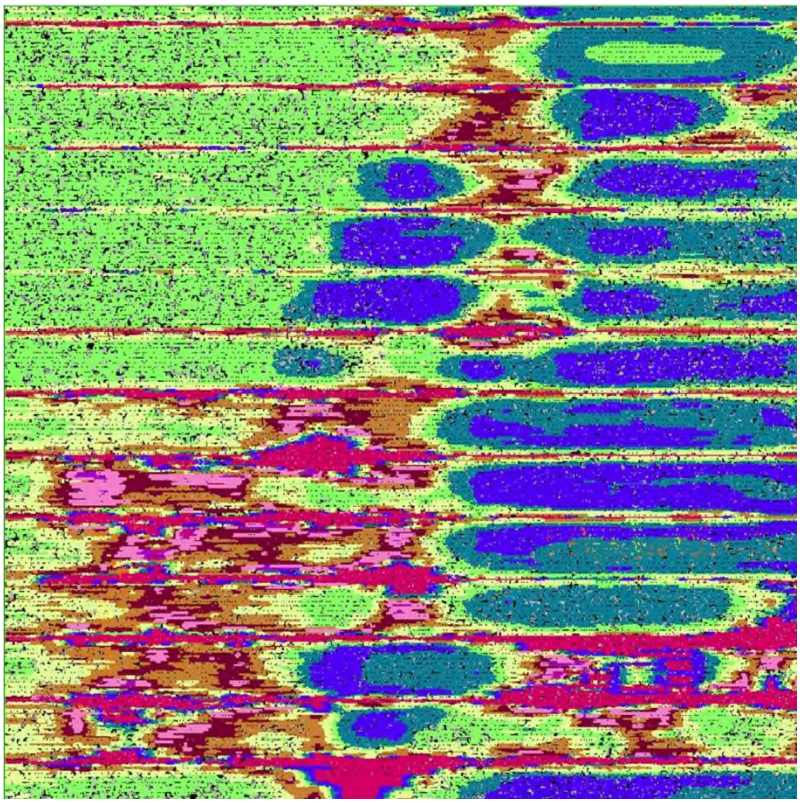
Timing Analysis and Power Integrity of Integrated Circuits in Technologies below 60nm.



Tsiampas Michail

Acknowledgements

I would like to express my gratitude to my PhD advisors, Professor George Stamoulis and Assistant Professor Nestoras Evmorfopoulos, for their endless support during all these years. I, wholeheartedly thank them, not only for their academic support and scientific guidance, but also for giving me so many wonderful opportunities. Being very different characters, but at the same time outstanding personalities they have managed, working complementary to create a powerful team, bringing extraordinary results. After all these years of cooperation, and all the things we have been through, I consider them to be my family.

A great thank you to HELIC and I am especially grateful to my CEO, Yorgos Koutsoyannopoulos who believed in me, gave me the opportunity to chase my dreams and at the same time created the necessary conditions for me to be able to fulfill both my goals, professional growth and conclude my PhD thesis. He was always there as a mentor.

I would also like to thank my good friends and colleagues Konstantis Daloukas, Errikos Lourandakis and Konstantinos Nikelis for the amazing cooperation during the past years and their essential assistance both at a technical and human level.

Finally, I want to thank my parents from the bottom of my heart, for always being on my side during my whole life, supporting my choices and teaching me never to quit. I dedicate this work to them.

It has been a long journey and an amazing experience !

1 Contents

2 Introduction

Technology Process continues to evolve, scaling down transistor sizes aiming to decrease power supply nominal voltages as the easiest way to lower the power footprint. At the same time modern deep submicron processes have stopped following Moore's law in voltage thresholds, reducing the gap of operation for each cell in the IC. Moreover reductions of transistor and conductor sizes lead to a proportional increase of the resistance of the metal layers, especially on the lower metals layers. Industry gradually moves towards to the production of Multi-core, Multi-die and Multi-GHz designs, which implies larger ICs, operating in even higher frequencies. The size of modern ICs, both in terms of number of elements and size of Power Supply Network, along with the simultaneous switching of its elements in high frequencies and the larger impedance of the Power Supply Network, aggravate the Power Supply Noise (Voltage-drop/IR-drop) during operation of the IC. Voltage-drop effect has become dominant nowadays, making designers wonder whether the voltage delivered to the design cells, is enough in order to be functional. Small changes in voltage can cause exponential changes in delays which can cause timing issues, unless there is a method to have timing be aware of the voltage conditions. The coupling of Voltage-drop and Static Timing becomes a cornerstone of electrical signoff. Even paths reported with fair timing, using Static Timing Analysis may be very sensitive to voltage fluctuation, thus there may be an input pattern for simulation which will cause that sensitivity to show up and generate a timing violation.

In EDA, analysis always targets the worst case conditions. Power Integrity analysis needs activity for the IC under test, which comes either from vectorless or vector based methodologies in order to find worst case Voltage-drop and max power consumption. On the other hand traditional Timing Analysis fails to capture the impact of Voltage-drop without being over pessimistic and even some times close to the real critical path. Considering the complexity of modern ICs, the number of all possible input patterns and the strongly coupled, interdependent dynamic effects taking place during simulation, finding the input pattern, which will lead to the worst case voltage-drop and consequently worst case delay in the IC is practically impossible, constituting a problem which cannot be solved analytically.

New methodologies are presented both for Power Integrity and Timing analysis of modern ICs in low technology nodes. The methodologies comply with all industrial standards (file formats and tools). Power Integrity analysis consists of a very fast and extremely accurate functional simulator with Power Supply Network simulation capability, tackling the problem of the tightly coupled effects of Voltage-drop and Timing, during the operation of the IC. The methodology estimates also the worst case voltage for all the cells of the design. The proposed Timing analysis methodologies have proven significantly more accurate than the existing methodologies in the field, introducing Voltage-drop aware Statistical Dynamic Timing Analysis using the results of the proposed Power Integrity methodology. For the statistical estimation parts of the methodologies, a powerful Statistical Prediction Engine was employed in two software implementations.

3 Design Constraints

There are three basic design constraints, Power, Timing and Area. All three types are highly correlated and changing one of them will affect the other two. The correlation though between Power and Timing is very strong and in modern ICs is of the essence since to meet the tough industrial specification. During the design cycle of a chip, there is always a trade-off between speed, power, area and all of them have to meet the initial constraints set by the designers. If one could set the importance of these design constraints, Timing would be the most important. Several EDA tools exist in the market aiding designers to estimate which will be the Power consumption, the performance and the area of a specific design after the Tape-out, which is the final result of the design process. Of course the design should properly work under extreme circumstances both regarding the physical environment (e.g. temperature) and technology process variation. This Thesis studies and presents new methodologies for the analysis and the estimation of the exact and worst case performance of moderns ICs in terms Power and Timing.

4 Design Cycle

The design cycle of an IC is a process starting from the specification (Design Constraints) of an IC and ending to the final tape-out of the design. This procedure might be iterative or partially iterative in the where the initial tape-out does not meet the specification. This complex and exhaustive process involves three major sub-processes the design of the IC, the manufacture and the test of the design. Main contributors are design and verification teams, IP vendors, and IC manufacturers. The individual steps taking place during an IC's design cycle can be described as follows:

1. Design Specification / Design Constraints:

   The specification usually includes the detailed requirements in terms of functionality related to the IC. Some of the basic requirements are the following

   - minimum operating frequency.
   - maximum power.
   - maximum area of the IC.
   - other robustness and reliability thresholds.

   The specification may come either from a company to a design-house or from the design-house aiming to sell the IC later on.

2. Architectural Design:

   A design architect is responsible to come up with a detailed specifications plan for each and every sub-block of the design, deriving information from the top-level specifications of the design.

3. RTL Design:

   Specific teams, which are called, design teams; take the partitioned design and using HDL (Verilog/SystemVerilog) describe the functionality of the design. The whole design is modeled at a higher level of abstraction using mixed style of modelling (behavioral, data-flow and structural).

4. Functional Verification:

   The functional verification is the procedure of developing a verification environment using code in HVL (System Verilog) in order to check whether the RTL design functions according to the specifications. In case of any errors, the design code is corrected and re-verified.

5. Logic Synthesis:

   Logic Synthesis is the process which converts the behavioral description of an IC (RTL code) into structural gate-level description. The behavioral description of the IC once it is tested and verified in terms of functionality as per the requirement, it is used as input to an EDA tool (logic synthesizer) which converts the description into a list of input/output ports (I/O of the design), standard cells with a pre-specified logic behavior and the signal connectivity between them (gate-level netlist).

6. Design-For-Test:

   This step makes certain changes, like adding extra logic, to the obtained netlist from the synthesis tool which will later aid in applying tests to verify the hardware.

7. Logical Equivalency Checking:

   Since Design-For-Test changes to the design, the gate-level netlist is yet to be verified. To avoid performing once again functional verification for this gate-level netlist, a tiresome task, it is faster to compare the verified, to the required extent, RTL code with the gate-level netlist by reducing both of them to Boolean equations and equating them to verify the gate-level netlist. This is logical equivalency checking and if both prove to be equal then our gate-level netlist is good to go.

8. Pre-layout Static Timing Analysis:

   After functional verification, designers have to verify whether the design operates at the required speed. This step exhaustively checks every timing path in the design for any setup and hold violations when the clock operates at a specified frequency. In case of any violations, the design should be fixed at this point before moving forward to the next step. Front-end part of the design ends here!

9. Floor Planning and Power Planning:

   The required area of the chip die, the core area within the die where it is permitted to place the synthesized design, area for blockages where no cell should be placed are determined during floor planning. Power Planning, determines how the power (VDD and GND) is going to be routed to the whole design.

10. Placement:

    During this step EDA tools, driven by experienced designers, calculate (decide) based on placement algorithms, the actual locations of the instances of the standard cells to be placed in the core area of the die are determined for the entire design. The best placement of the instances in the area

constitutes an NP-Complete problem which cannot be solved analytically. Thus heuristic methods are incorporated in the algorithms.

11. Clock-Tree Synthesis:
    Goal of this step is to ensure that the active edge of the clock signal reaches every flip-flop in the design at the same time.

12. Global Routing:
    The global routing determines the paths through which the cells are to be connected on a 2D plane.

13. Detailed Routing:
    The detailed routing determines the exact path the interconnections take on a 3D space through various metal layers and vias along the path chosen by global routing.

14. Physical Verification and RC Extraction:
    The placed and routed design is checked for the presence of any Design Rule violations (DRC - If the layout meets every design rule given by the foundry for the given technology), Layout Vs Schematic (LVS) mis-matches, Electrical Rule violations (ERC), XOR check, Antenna Check etc. The RC parasitics for all the components in the design are extracted.

15. Post Layout Static Timing Analysis:
    Now, with all parasitics extracted and the actual clock parameters, wire and cell delays known, the STA is performed once again with the parastics annotated, to verify the timing correctness of the design exhaustively. STA though, is usually performed at several stages during the back-end design flow, which is consolidated as a single step here. Back-end design ends here!

16. GDSII generation:
    The GDSII (Graphic Database System for Information Interchange) is a format which is understood and used by the foundry for fabrication of an IC. After the back-end flow, the GDSII stream is generated from the final layout and taped-out to the foundry.

17. Post-Silicon Validation and Testing:
    After the IC is manufactured, several sample-chips from the wafers are again tested for presence or absence of any possible faults during the fabrication process. The chips are validated against the initial specifications given by the customer whether all the specs are met or not. The chips which turn out to be fault-free and meeting all the specs, come to the market.

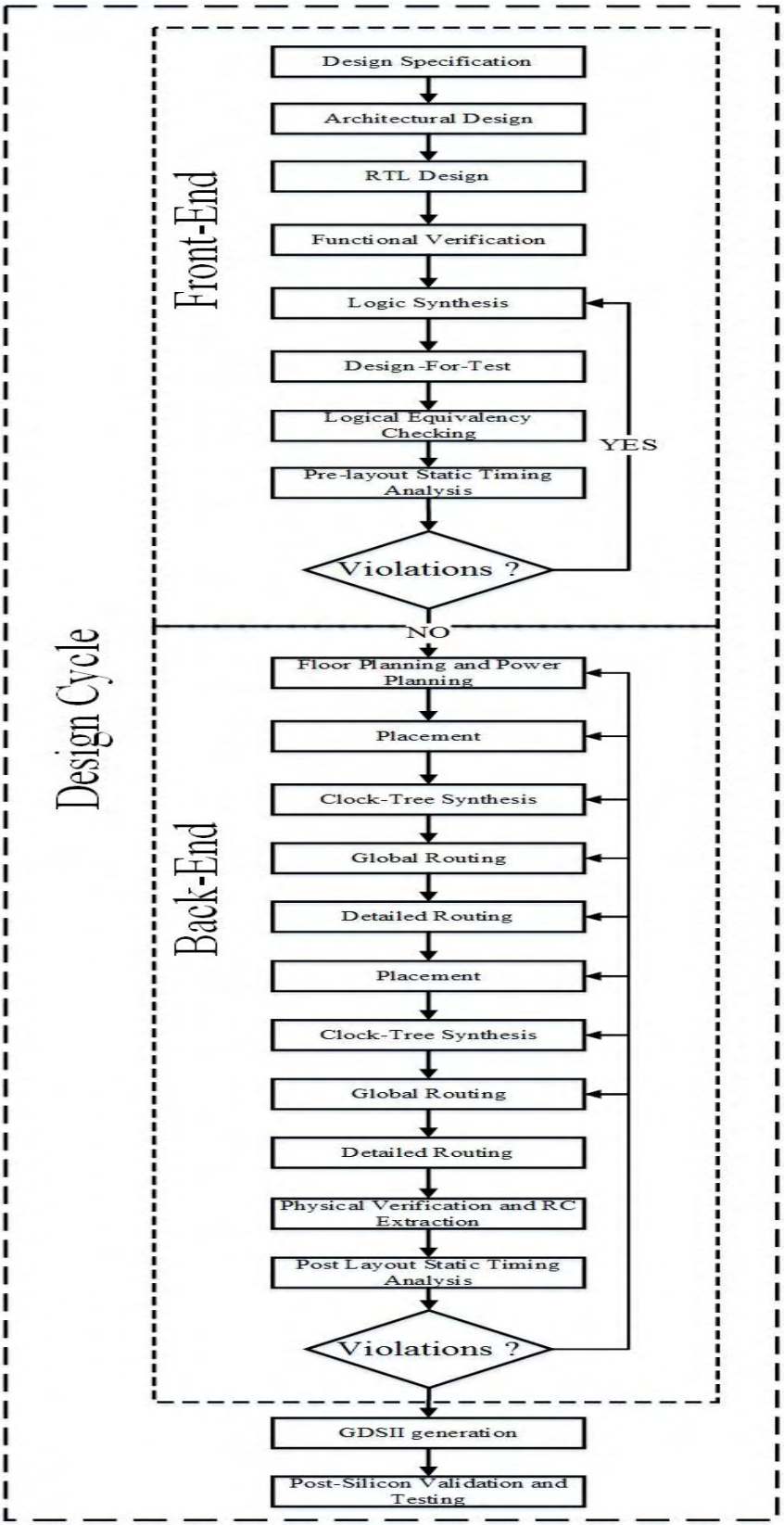


Figure 1 Design Cycle flow.

5 Power Integrity

5.1 General description

Power Integrity (PI) is an integral part of the Back-end of the design cycle of an IC and constitutes an analysis reporting whether the lower voltage levels along with the maximum current consumption reported by the corresponding PI-EDA tools for an IC under test exceed or not, the thresholds defined in the initial specification which includes the IC requirements. In other words, PI calculates the voltage level and amount of current reaching the devices and checks if these are sufficient for the proper functionality of the IC. To do so, the Power Supply Network has to be modeled as an RLC network and then simulated using the power pins of the devices and the power pads/bumps as ports of this RLC network. Nowadays, power integrity analysis is a key factor to a successful product and its report determines whether it is safe for a design team to proceed to the Tape-out process or not. Power integrity analysis is tightly coupled with Timing analysis which is also a subject of the current thesis.

5.2 Power Supply Network Analysis

The Power Supply Network is the set of all the interconnected metal wires and vias, comprising a net which is responsible for delivering the necessary amount of current from the IC's power pads or power bumps, which are connected to the external power sources, to all elements that together compose the functional logic part of an IC. The external power sources are connected in a distributive way to several tap-points of the Power and Ground Supply Networks (Power Distribution Networks - PDN) through the upper most layers of the technology stack, known as the package layers. This connection between the package of the IC and the Power Distribution Networks is done either using wire-bond method (an old but stable chip technology) or using C4 bump arrays which is a modern chip technology. Continuously shrinking device dimensions in modern deep sub-micron technology processes, along with the continuously increasing and more complex multifunctional modern ICs lead to high switching activities and thus to significantly higher power consumption. In these modern and power hungry ICs the ability of delivering large amount of currents to the devices within an acceptable time slot is of high importance. The aforementioned description can be translated to the ability of the power supply network to preserve an acceptable voltage level, close to the voltage level of the external power supply, on the power pins of all devices in the design during its operation. The relationship between voltage and current is expressed through the well-known equation V=I*R. Since the current drawn by the devices depends on the switching activity of each cell and the device parameters of the technology process, the voltage level over each device of the IC is regulated by the impedance mediates from the external power supply to the voltage pin of the devices. This impedance is the one induced by the only connection between these points of the design which are the Power and Ground Supply Networks. Due to the resistance induced by the interconnected wires, which constitute the power delivery network, there is a voltage drop across this network. This is known as the IR-drop or Voltage-drop effect and the losses induced from it are referred to as the resistive losses I*R of the network. Although the package of the design on the one hand does not induce significant resistance, on the other hand it induces significant inductance to the power supply network. The slow response of the inductance to fast changes of currents in the Power Supply Network creates the L*di/dt drop effect or inductive losses. The sum of inductive losses and resistive losses (I*R+ L*di/dt) is responsible for the total impedance losses inducing the voltage drop effect (IR-drop). The performance of an IC is strongly related to the ability of its devices to switch a state fast, which in turn is strongly related to the supplied voltage seen on the devices, is the subtraction of the external voltage supply and the Voltage-drop. High voltage drops on the

power and ground delivery networks result to reduced switching speeds and noise margins of an IC, and the induced noise may lead to functional failures. Moreover high current densities may lead to undesirable wearing out of interconnect wires due to electromigration (EM) effect. As one may infer the robustness of the power and ground supply networks is essential for the reliability of modern high performance ICs. The challenges of designing a robust power supply network is to achieve delivering sufficient voltage level at the power pin of the devices and absorbing the large fluctuations of the currents across the IC and use as smaller area of metal layers as possible. Due to the large scale of modern designs and the high power consumptions a large hierarchy of metal layers is used to distribute this large amount of power across the design. To increase the ability to deliver sufficient voltage level to devices and the tolerance to current fluctuations several design technics are introduced during the Power Supply Network design. The most common technic is to connect capacitances, between the power and ground supply networks, at strategic locations of the chip, commonly referred to as decoupling capacitors or Decaps. These capacitors are used a local charge storages and help to mitigate the voltage drop effect at supply points. Unfortunately this technic is not always enough to limit the voltage drop under an acceptable threshold. Moreover, this technic introduces extra leakage power consumption to the design and also increases the IC area. The crux in designing a robust power supply network is the large number of unknown factors existing until the final steps of the design cycle. Even though these unknown factors exist, the structure, the size and the layout of the power and ground supply network have to be decided by the designers at the very early stages, when the largest part of the IC design, has not even begun. Most of the industrial tools used for power grid verification are built in order to be used at the final stages of the design cycle where the IC design and thus the power grid design is complete and extracted to a parasitic network. This fact introduces extra difficulty in fixing issues revealed on the power grid. The main problem on in power supply network analysis is the extremely large size of the networks. Typically their size reaches millions of nodes in modern deep submicron designs. The simultaneous simulation of both the power/ground supply networks and the digital part is computationally infeasible. In order modern EDA tools to be able to handle the size of this type of simulation splits the process in two separate steps. The first step is the simulation of the digital part which performs the calculation of the currents drawn by the digital elements during IC operation assuming ideal voltage level on their power and ground pins. The second step is the simulation of the power and ground supply network modeling the devices as independent time-varying current sources connected to the power grid. This approach ignores the interaction and the aforementioned relationship between the performance of digital devices and the voltage level they “see”. This is a simplification of the operation of an IC making easier for the designers to test the power grid but at the same time this simplification introduces significant deviation between the simulation results and the real-world behavior of the power grid of the IC.

The switching activity and thus the power consumption of a design are tightly correlated to the input patterns used for the simulation. The analysis of the power supply network is classified in two different types, the vectorless and the vector dependent methods. The input dependent methods search for the input vector set which produces the worst-case voltage drops in the IC while the vectorless methods try to find an upper threshold of the worst case voltage drop in an efficient way. The input pattern dependent methods used mainly genetic algorithms trying to find input vectors or pairs of input vectors that maximize the power consumption in the design. This approach tries to solve an NP-Complete problem underestimating the impact of voltage in design performance and thus leaving problems in power delivery network unnoticed. Vectorless methods are fast but too conservative leading to overdesigning the power supply network.

5.2.1 Power Supply Network Parasitic Extraction

The Power Supply Network (*Power Delivery Network/PDN/Power Grid*) of an IC is a mesh of metal layers (wires) connecting the Power Pads or Power Bumps (C4 - Bumps), which are the connection ports with the external power supply for the IC, to the power pins of its elements. The elements of an IC may be standard cells, memory blocks or any other type of digital or analog devices. The elements of an IC are designed to be functional within a pre-specified range of voltage thresholds. Thus the Power Grid is essential for the proper functionality of the design. In order for the designers to be able to verify the proper functionality of the design, a check they are responsible for, is whether the voltage level over the Power pin (VDD pin) of each element of the design lies within the specified voltage boundaries. To do so the Power Grid has to be modeled in a valid simulatable file format in order to be used as input to existing industrial simulation tools. As a result, the Power Grid is modeled as a complex network consisting of resistances, capacitors and inductances. This modeling is also known as parasitic extraction of the power grid. The tools which perform this type of functionality are called parasitic extraction tools. The basic inputs for a parasitic extraction tool are:

1. The design in GDSII or DEF file format.
2. The technology parameters of the PDK (Process Design Kit) that the IC is designed.
3. Configuration files with several parameters setting the mode of Parasitic Extraction Tool.

The following two figures are a very simple example explaining how the power grid of a design is modeled. In **Figure 2**, the connected metal wires (blue, green, orange lines) comprising the Power Supply Network, are connected to the power pins of a design.

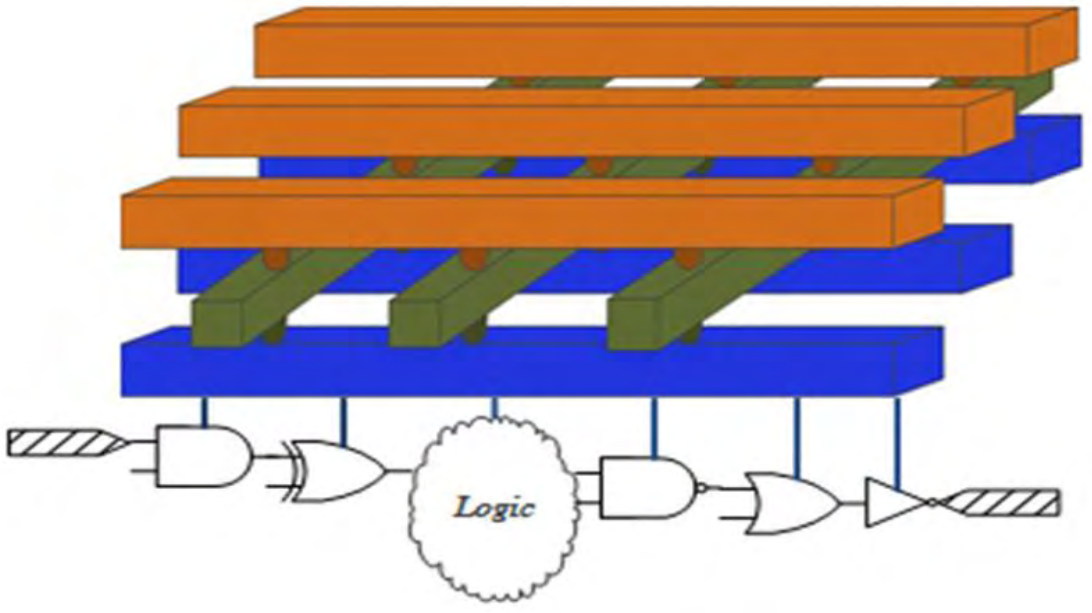


Figure 2 Power Supply Network and IC connectivity.

The **Figure 3** depicts the results of the Power Supply Network extraction process of **Figure 2**. The network of metal wires is now modeled as a network of connected resistors, capacitors and inductors (parasitic extraction passive elements). The orange metal wires correspond to the orange elements of the extracted network in **Figure 3**. The devices (the std-cells in our case) are modeled as current sources while the power sources correspond to power pads or power bumps.

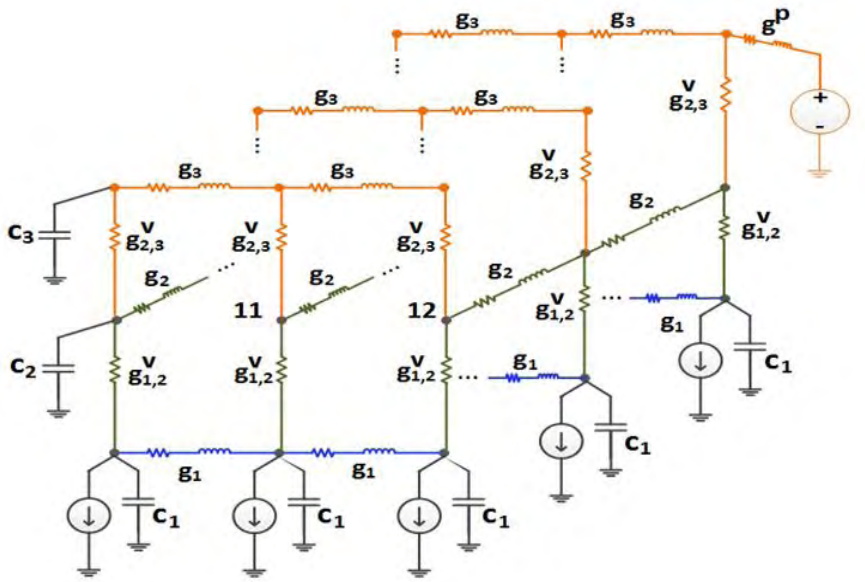


Figure 3 Power Supply Network parasitic model.

The larger and more complex a Power Supply Network is, the larger and more complex the extracted result will be. Thus several mathematical methods (numerical reduction methods) have been developed lately in order to reduce the result of the parasitic extraction process. Indicatively an average sized modern IC, of an area ~1.8 sq. mm, designed with 28nm Technology Process node, has a Power Supply Network modeled with passive elements of the following numbers:

- Number of nodes : 600 K
- Number of inductors : 400 K
- Number of resistors : 1.3 M
- Number of capacitors : 1.5 M

5.2.2 Power Supply Network Simulation

Efficient analysis of massive on-chip power delivery networks is among the most challenging problems facing the EDA industry today. The on-chip power delivery network constitutes a vital subsystem of modern nanometer-scale ICs, since it affects in a critical way the performance and correct operation of the devices. In order to determine the quality of the supply voltage delivered to the devices, the designer has to perform static and dynamic simulation of the electrical circuit modeling the power grid. This has become a very challenging problem for contemporary ICs, since power grids encountered in these circuits are extremely large (comprising several thousands or millions of nodes) and very difficult to simulate efficiently (especially over multiple time-steps). Static (DC) or transient simulation refers to the process of computing the response of an electrical circuit to a constant or time varying stimulus. Since a power delivery network can be generally modeled as a linear RLC circuit, the process of DC or transient simulation of large-scale power grids amounts to solving very large (and sparse) linear systems of equations. Direct methods (based on matrix factorization) have been widely used in the past for solving the resulting linear systems, mainly because of their robustness in most types of problems. They also have the property of reusability of factorization results in transient simulation with a fixed time-step. Unfortunately, these methods do not scale well with the dimension of the linear system, and become prohibitively expensive for circuits beyond a few thousand elements, in both execution time and memory requirements. In addition, a fixed time-step is almost never used in practice because it becomes very inefficient to constantly simulate during long intervals of low activity. All practical implementations of integration techniques for ordinary differential equations (ODEs) employ a variable or adaptive time-step mechanism [1]. In such cases, the reusability of matrix factorization in direct methods ceases to exist.

The general form of a linear system of equations is described with the following equation:

$$(5.1)\quad Ax = b$$

Where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a n x n matrix of real numbers , $\mathbf{b} \in \mathbb{R}^{n}$ is a vector of size n, and $\mathbf{x} \in \mathbb{R}^{n}$ is an unknown solution vector of size n that will be determined by a solution method. A solution for the linear system in (5.1) exists only if the matrix A is non-singular, which means that the inverse matrix $\mathbf{A^{-1}}$ with $\mathbf{AA^{-1}} = \mathbf{I}$ exists. The solution methods for the linear systems are classified as **direct** or **iterative**. The direct methods solve the above linear system in a predefined number of steps, which depends on the size n of the linear system. The iterative solution methods determine an approximation of the exact to a predefined accuracy level. The symmetry and the positive definiteness are two key properties of a matrix and allow utilization of more efficient direct or iterative methods for the solution of the corresponding system.

5.2.2.1 Direct Solvers

Direct solvers solve the linear system in (5.1) in a predefined number of steps, which depends on the size of the linear system. They consist of two steps, namely a factorization step where the system matrix is decomposed into a number of factors, and the solution phase where the matrix factorization is used for the solution of the initial system. LU factorization is the direct method used in general, non-symmetric matrices. It factors the system matrix in two factors, one lower- and one upper-triangular matrix $A = LU$, and the equation (5.1) is transformed into the following:

$$(5.2)\quad Ax = b \Rightarrow (LU)x = b \Rightarrow L(Ux) = b$$

Where **L** is a lower-triangle matrix and **U** is an upper-triangular matrix. As a result, the original system is transformed into two equivalent systems and is solved into two steps as follows:

$$(5.3)\quad Ly = b$$

$$Ux = y \quad (5.4)$$

The advantage of breaking the original system into the above set is that each linear system requires the solution of a triangular system (forward substitution for the first system and backward substitution for the second one) which is a trivial computational process. The main advantages of direct methods are their robustness and the fact that once the factorization is complete, the solution of the linear system, even with multiple right hand side vectors, is a trivial process as long as the system matrix remains the same. However direct solvers present superlinear scaling both in computational and memory requirements with the increasing size of the linear system, which rules them out for large-scale linear systems.

5.2.2.2 Iterative Solvers

The relentless push for high-performance and low-power integrated circuits has been met by aggressive technology scaling, which enabled the integration of a vast number of devices on the same die but brought new problems and challenges to the surface. The on-chip power delivery network (power grid) constitutes a vital subsystem of modern nanometer-scale ICs, since it affects in a critical way the performance and correct operation of the devices. In order to determine the quality of the supply voltage delivered to the devices, the designer has to perform static and dynamic simulation of the electrical circuit modeling the power grid. This has become a very challenging problem for contemporary ICs, since power grids encountered in these circuits are extremely large (comprising several thousands or millions of nodes) and very difficult to simulate efficiently (especially over multiple time-steps).

Iterative methods involve only inner products and matrix-vector products, and constitute a better alternative for large sparse linear systems in many respects, being more computationally- and memory-efficient. This holds even more so for modern nonstationary iterative methods which fall under the broad class of Krylov-subspace methods [2]. Iterative methods possess themselves a kind of reusability property for transient simulation, in that the solution in the last time-step provides an excellent initial guess for the next time-step, thus making a properly implemented iterative method converge in a fairly small number of iterations. In fact, this property also holds in the case of a variable time-step, since the quality of the last solution as initial guess for the next solution is not affected. The above features make iterative methods much more suitable for DC and variable time-step transient analysis of large-scale linear circuits such as power distribution networks.

In general, iterative methods belong to the category of relaxation methods. They start using an initial solution guess, providing a partial solution in each time step and at the end the algorithms converge to the final solution, within a predefined accuracy. The two categories of iterative methods are the following:

1. The Stationary methods, which solve the linear system using an approximation matrix of the original one, using a series of steps trying to minimize the error of the result. The initial matrix is usually decomposed generating the approximation matrix allowing for more accurate solution. The most well-known stationary methods are the *Jacob*i, the *Gauss-Seidel* and *Successive Over Relaxation (SOR)*. If (**D**) is the diagonal, (**U**) the upper triangular and (**L**) the lower triangular parts of the decomposed system matrix **A = D + L + U**, the approximation for each of the aforementioned iterative method is as follows:
    a. *Jacobi*: $x^{(k+1)} = \mathbf{D}^{-1}(\mathbf{b}-\mathbf{R}\mathbf{x}^{(k)})$, where $\mathbf{R} = \mathbf{U} + \mathbf{L}$.
    b. *Gauss-Seidel*: $x^{(k+1)} = \mathbf{L}_*^{-1}(\mathbf{b}-\mathbf{U}\mathbf{x}^{(k)})$, where $\mathbf{L}_* = \mathbf{L} + \mathbf{D}$.
    c. *SOR*: $x^{(k+1)} = (\mathbf{D} + \omega\mathbf{L})^{-1}(\omega\mathbf{b} - (\omega\mathbf{U} + (\omega-1)\mathbf{D}))x^{(k)} = \mathbf{L}_\omega x^{(k)} + \mathbf{c}$, where $\mathbf{L}_\omega = -(\mathbf{D} + \omega\mathbf{L})^{-1}(\omega\mathbf{U} + (\omega-1)\mathbf{D})$, $\mathbf{c} = (D + \omega\mathbf{L})^{-1}\omega\mathbf{b}$, and $0 < \omega < 2$.

2. The Non-stationary (or Krylov-subspace) methods, forming the *Krylov* sequence which is a basis of sequence of successive matrix power times the initial residual. The method then forms the approximation by minimizing the residual over the subspace formed. The most well-known *Krylov*-subspace methods are the Conjugate Gradient (CG), the Generalized Minimal Residual Method (GMRES) and the Biconjugate Gradient Method (BiCG). The CG method is applicable to SPD systems while GMRES and BiCG are applicable on non-symmetric problems.

For the linear systems used in Power Supply Network analysis, the SPD matrices are considered to be the most appropriate.

5.2.2.2.1 Conjugate Gradient Method

The CG method is the first Krylov-subspace iterative method that was developed for SPD matrices. The base of the algorithm is the theory of global minimization and orthogonal polynomials. The algorithm's goal is to minimize through iterations for the x_i the A-norm:

$$(5.5)\ \|x_i - x\|^2 = (x_i - x, \boldsymbol{A}(x_i - x))$$

that are in the Krylov subspace $K_i(\boldsymbol{A}, r_0) \equiv \{r_0, \dots, A^{i-1} r_0\}$. The solution of the linear system is then approximated by the CG method which computes a series of residual vectors. The current residual vector of each time step is orthogonal to the space of the previously generated residuals. The solution vector of the final iteration approximates the real solution of the initial system within the user's predefined accuracy level. As for the convergence rate of the method, it can be shown that the required number of iterations (for a given initial guess and convergence tolerance) is bounded in terms of the spectral condition number $k_2(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \left\|\boldsymbol{A^{-1}}\right\|_2 \geq 1$ specifically; it is $O(\sqrt{k_2(\boldsymbol{A})})$, which for SPD matrices becomes $k_2(\boldsymbol{A}) = \frac{\lambda\_\max(\boldsymbol{A})}{\lambda\_\min(\boldsymbol{A})}$ where $\lambda_{max}(\mathbf{A})$, $\lambda_{min}(\mathbf{A})$ are the maximum and minimum eigenvalues of **A** respectively. This means that the convergence of the method, becomes fast when $k_2(\boldsymbol{A}) \cong 1$ and slow when $k_2(\boldsymbol{A}) \gg 1$. The main drawback of the method is the unknown convergence rate which depends on the spectral properties of the matrix **A** of the linear system. A process known as preconditioning is used to transform the original matrix to a matrix with more favorable properties.

5.2.2.3 Voltage-Drop Effect

In Section 5.2.1 we have seen that the Power Supply Network of an IC is modeled as a complex RLC network. The RLC elements introduce complex impedance from the power supply nodes of the network (Power Pads/Power Bumps) to the power supply pins of the IC's devices. This complex impedance introduces the Voltage-drop effect in the IC. Voltage-drop (IR-drop/Power Supply Noise) is a terminology used to describe the way power, which is supplied by external voltage source, is reduced, as the current drawn from the digital devices flows within the passive parasitic elements of an passive circuit. Voltage-drops across internal resistances , conductors, contacts, and connectors are undesired as the supplied energy is lost (dissipated). **Figure 4** illustrates the Voltage-drop effect and the impact of the complex impedance of the PDN on the voltage level over the power pin of the gates. The Power Supply Network is modeled as a simple R only resistive net connecting the power supply source of 1.0V, to the power pin (node n1) of an inverter. As we can see the current waveforms are identical both on node N1 and node N2, whereas the voltage waveforms on nodes N1 and N2 are different. To be more precise, the lowest point of the voltage waveform on N2 is lower than the lowest point of the voltage waveform on node N1. This reduction of the voltage from the ideal level, which in our example is 1.0V, to something lower over the power pin of the IC's devices (node n1 in our example) is called Voltage-drop effect.

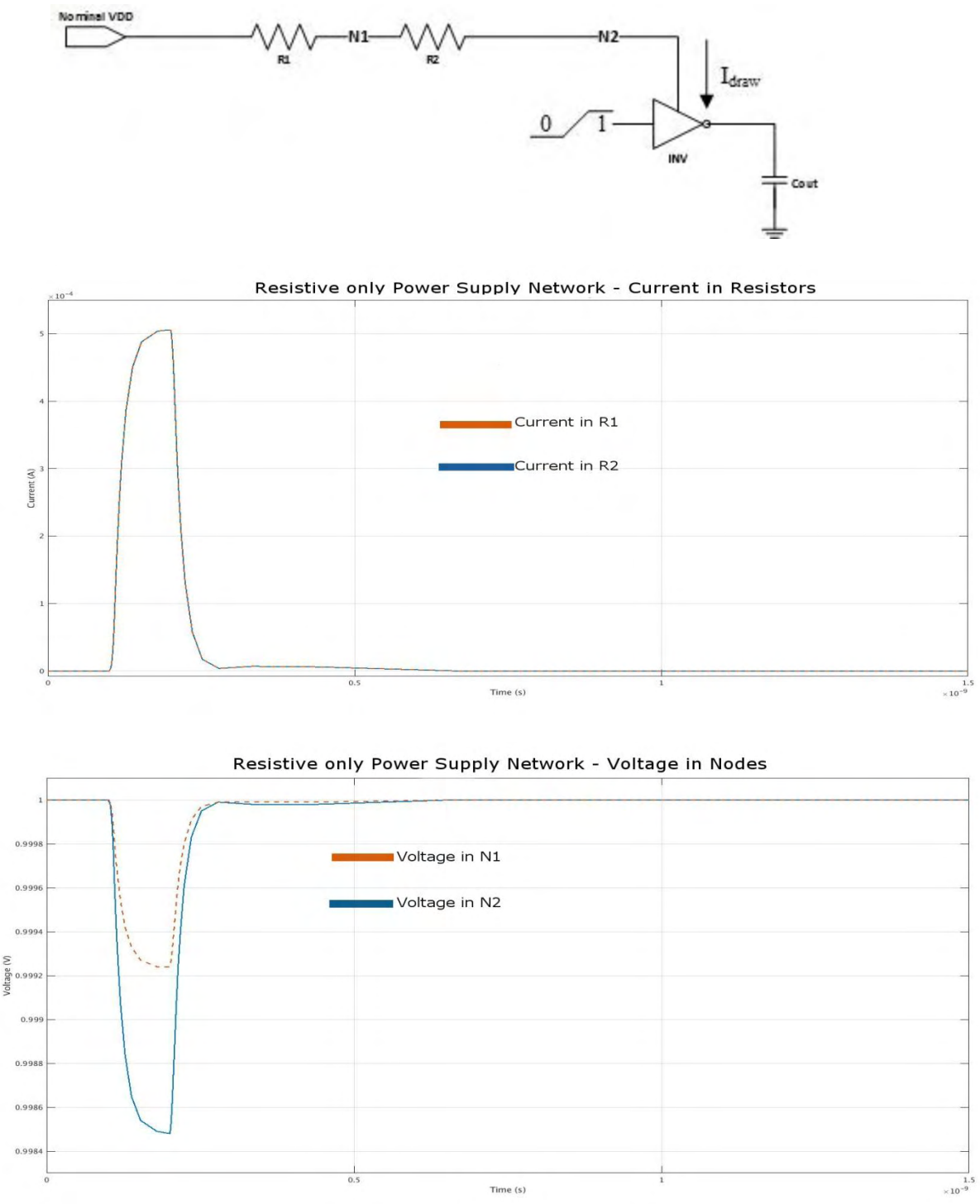


Figure 4 Voltage-drop Effect (R-only extraction).

In **Figure 5**, the Power Grid is modeled using additionally to the resistances, one capacitor (C1), thus the Power Supply Network is modeled as an RC network. The impact of this extra capacitor in the model is depicted, both in the current waveforms and the voltage waveforms of nodes N1 and N2. The extra capacitor

works as local charge storage and now the lowest point of the voltage waveform is better than before. Very interesting is the impact of this extra capacitor on the current waveform of node N2. The current waveform is now shifted in time and it is wider, which means that C_{out} will be charged with delay compared to the previous Power Supply Network modeling.

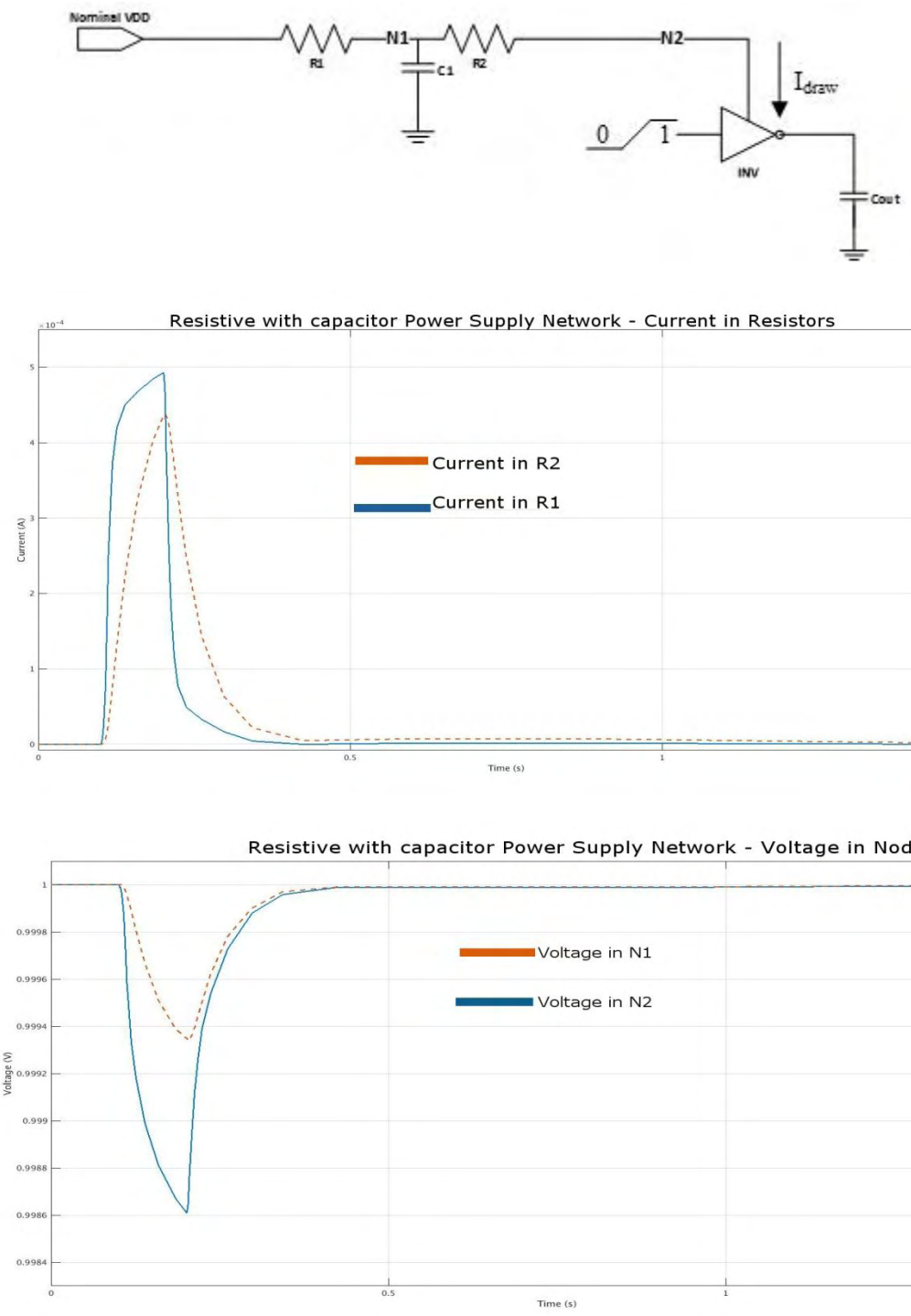


Figure 5 Voltage-drop Effect (RC extraction).

5.2.2.4 *Electromigration Effect*

Electromigration (EM) effect is very similar to Voltage-drop, sharing the same root of problem; high current densities running through very small, resistive metal wires. The increased level of current density result to the wearing out of metal wires. High current density forces the metal ions to move towards the current direction (**Figure 6**). This effect can be characterized by the ion flux density. The strength of the forces holding the ions in place, the conductor type, the size of the crystal, the temperature, the mechanical stresses, the grain-boundary chemical composition along with current density and the forces that tend to dislocate them, are the environmental variables from which this flux density depends on.

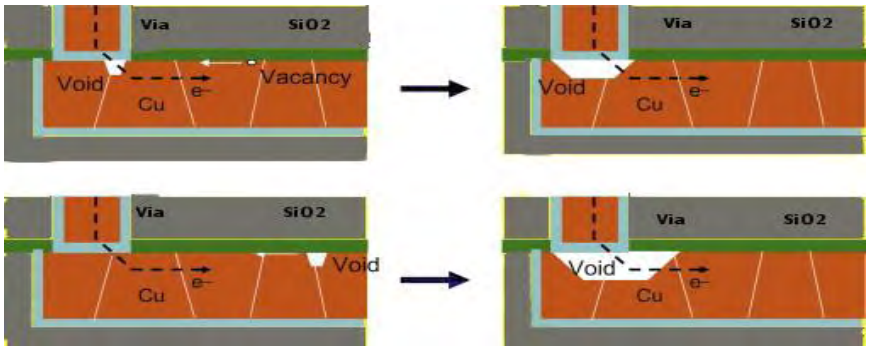


Figure 6 Example of Electromigration Effect [4].

While EM problem has been detected since 90nm or even earlier, it gets worse at advanced nodes such as 20nm and below. The reason for this is that wires are getting thinner and current densities become higher. As wires are scale down to more advanced process nodes, resistance increases and more current is needed. Similarly, in order designers to meet the performance specification of the devices, the current density has to increase. **Figure 7** depicts an image from electronic microscope, exposing the opens and voids due to EM effect on real IC power-pads on the left and power-bumps (C4) on the right.

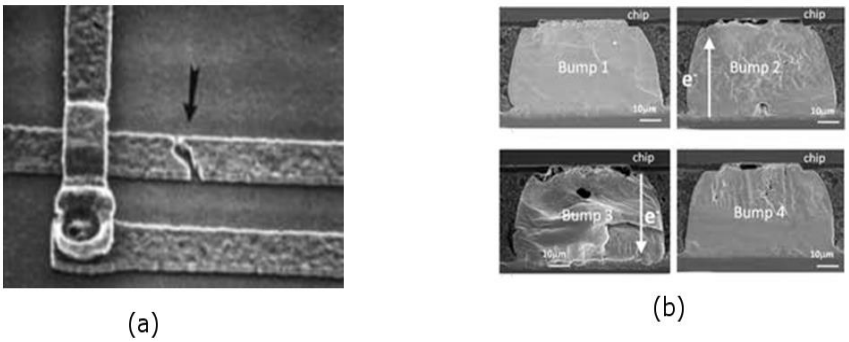


Figure 7 Wire [5] and Bump [6] open due to Electromigration Effect.

5.3 Existing Power Integrity Methodologies

The most common approaches in the industry, for IC Power Integrity can be described by the flow depicted in **Figure 8**. The main preprocessing steps, like Power Grid Extraction and library precharacterization are the same in all existing approaches and also mandatory. One may easily infer from the flow that existing approaches do not take into account the voltage over the std-cells in order to generate the current waveforms resulting from the IC's switching activity, which makes the current waveform wrong and thus the corresponding voltage waveforms also wrong or at least overly pessimistic.

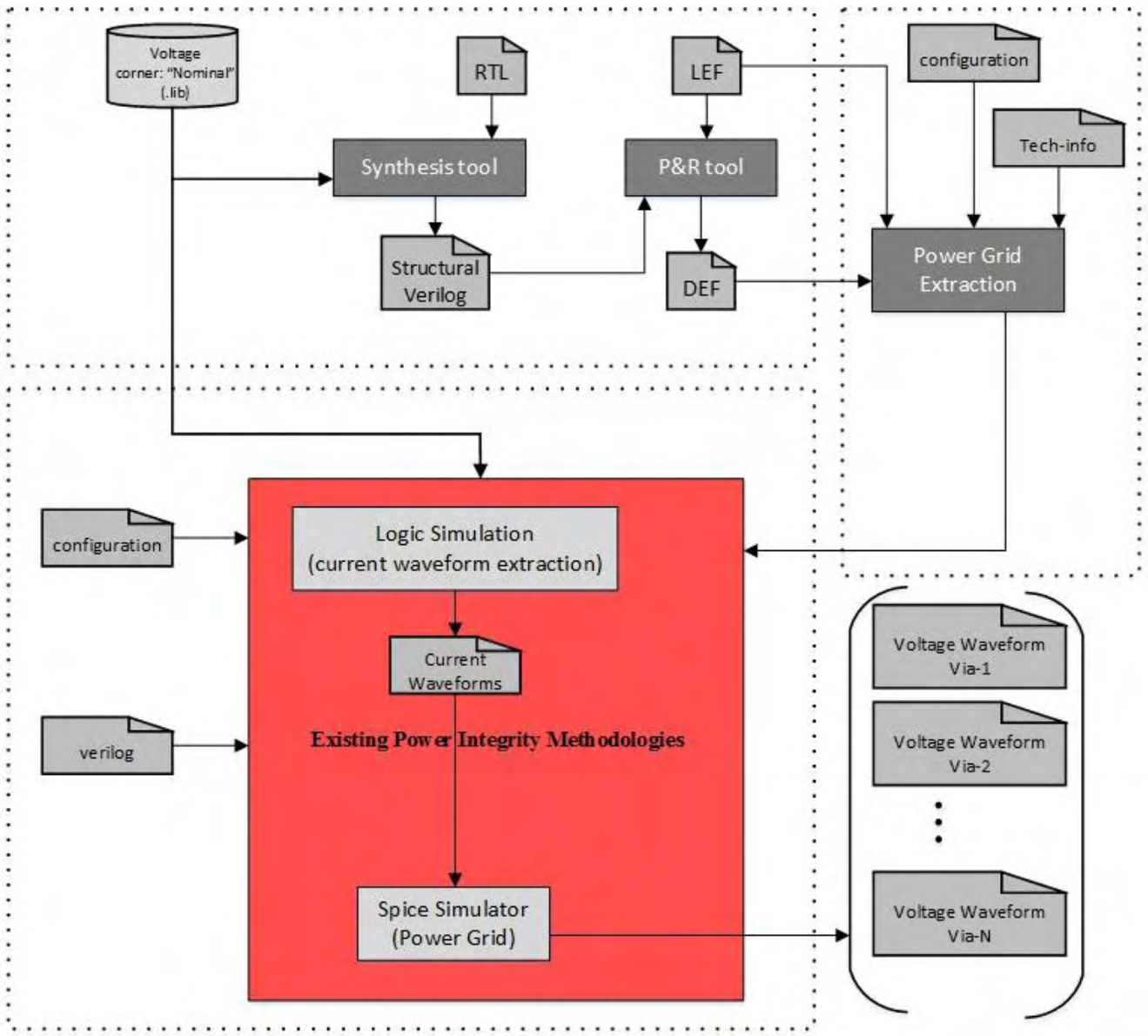


Figure 8 Existing Power Integrity methodologies

5.4 Proposed Power Integrity Methodology

In order to develop a holistic solution on the complex problem of the power integrity verification for modern deep submicron ICs, a novel methodology which simultaneously tackles both all individual and interdependent problems, should be introduced. This new methodology has to address these issues from a rather different perspective from the existing methodologies in the field, providing at the same time accuracy, speed and accurate prediction of the true worst case voltage drops at any tap point of the power supply network of the IC under test, without requiring from designers to determine a specific set of input vectors. A tool called NANOPOWER™ [60] has been developed, implementing this novel methodology, which is able to fulfill all the aforementioned specification from a designer's perspective.

The tool, in its final version, achieves these goals using a unique mix of three lynchpin technologies, (a) a high-capacity simulation engine, with grid awareness that makes use of pre-characterized cell libraries and produces SPICE-accurate rail current waveforms, (b) a fast linear solver, and (c) a statistical prediction engine based on solid mathematical foundation, capable of calculating the worst-case estimates at 99.99% confidence based upon a relatively small number of either internally generated or externally given simulation vectors or switching activity information. The basic inputs of the tool are the extracted power supply network of the place and routed IC, the structural Verilog describing the gate level netlist of the IC and the libraries in .lib format with the precharacterized cells. The cells must be precharacterized for several discrete values of power and ground supply voltages, in order for the tool to achieve a sufficient level of accuracy during simulation. Usually, four to six libraries in different levels of power and ground voltage are enough. Originally the tool has to accurately extract the power and ground supply network of the IC, which is implemented using HELIC's RaptorX™ tool in RC or RLC mode, generating a spice-format file with the extracted netlist and two additional files with information regarding the connectivity, of all the vias and the supply pins of the cells on the extracted netlist, respectively.

The proposed methodology has been evolved to its final version through two implementations of a tool, depicted in **Figure 8** and **Figure 9** respectively. An indicative design flow, including the aforementioned fundamental preprocessing steps, is also depicted outside the colored boxes. The cells have been precharacterized, in several voltage corners, building a set of libraries (in different voltage corner) that include the current waveforms drawn (CCS power model) and the delay (NLDM described in **Appendix A** or CCS timing model described in **Appendix B**) for all possible logic combinations on the inputs, for all std-cells of the PDK. The designs description is in structural Verilog format. The Verilog file is exported by a synthesis tool. The placed and routed design is in DEF (Design Exchange Format) file format and is exported from a Place&Route tool, using the structural Verilog from the previous step and the LEF (Library Exchange Format) file from the PDK. The power supply network of the Place&Routed IC is then extracted and its parasitic model, along with the auxiliary files, including the connectivity of the digital part to the power supply network, is exported. As we have already described in previous sub-sections of Power Integrity, these are all necessary steps, performed by all Power Integrity tools in the industry.

NANOPOWER™ introduces a new approach which eliminates the limitations resulting from the existing power integrity methodologies. Existing approaches in the industry use constant rail approximation for the calculation of the current and the voltage waveforms, considering the current waveforms generated by the switching activity of the logic simulation, independent from the power supply network simulation. The operation of the tool is configured from a configuration file, including all external input files, all reports, the mode of operation, the type of Linear Solver (Direct or Iterative) to be used, the extracted power supply network, the extracted

interconnects in SPEF format, the precharacterized libraries and several other options. In the end, the proposed methodology reports the average power consumption, the worst case voltage-drop waveforms for every via (tap-point) connecting M1 metal layers to the rest of the power/ground supply network, along with the confidence-interval waveforms describing the upper and lower bounds of the estimations (detailed description in chapter *Statistical Prediction Engine*). Finally the methodology reports three configuration files, to be used by Synopsys PrimeTime® for voltage-drop aware STA. These files contain all std-cells and macros of the design, having the real worst-case voltages the way they were estimated from the Statistical Prediction Engine. The first file contain the cells annotated with the worst-case voltage and the other two files the cells annotated with the upper and lower bounds of the estimation (the confidence interval values). In order for the methodology to reach spice accuracy on its results it performs iteration between the currents calculation module (EDS) and the Power Supply Network solver calculating the voltage waveforms.

In the first version of the tool (**Figure 9**) the Worst Case Voltage waveforms for each tap-point of the IC is an outcome of the simulation (LinearSolver) of the power/ground supply network using the worst case current waveforms as the latest were estimated by the Statistical Prediction Engine of the tool.

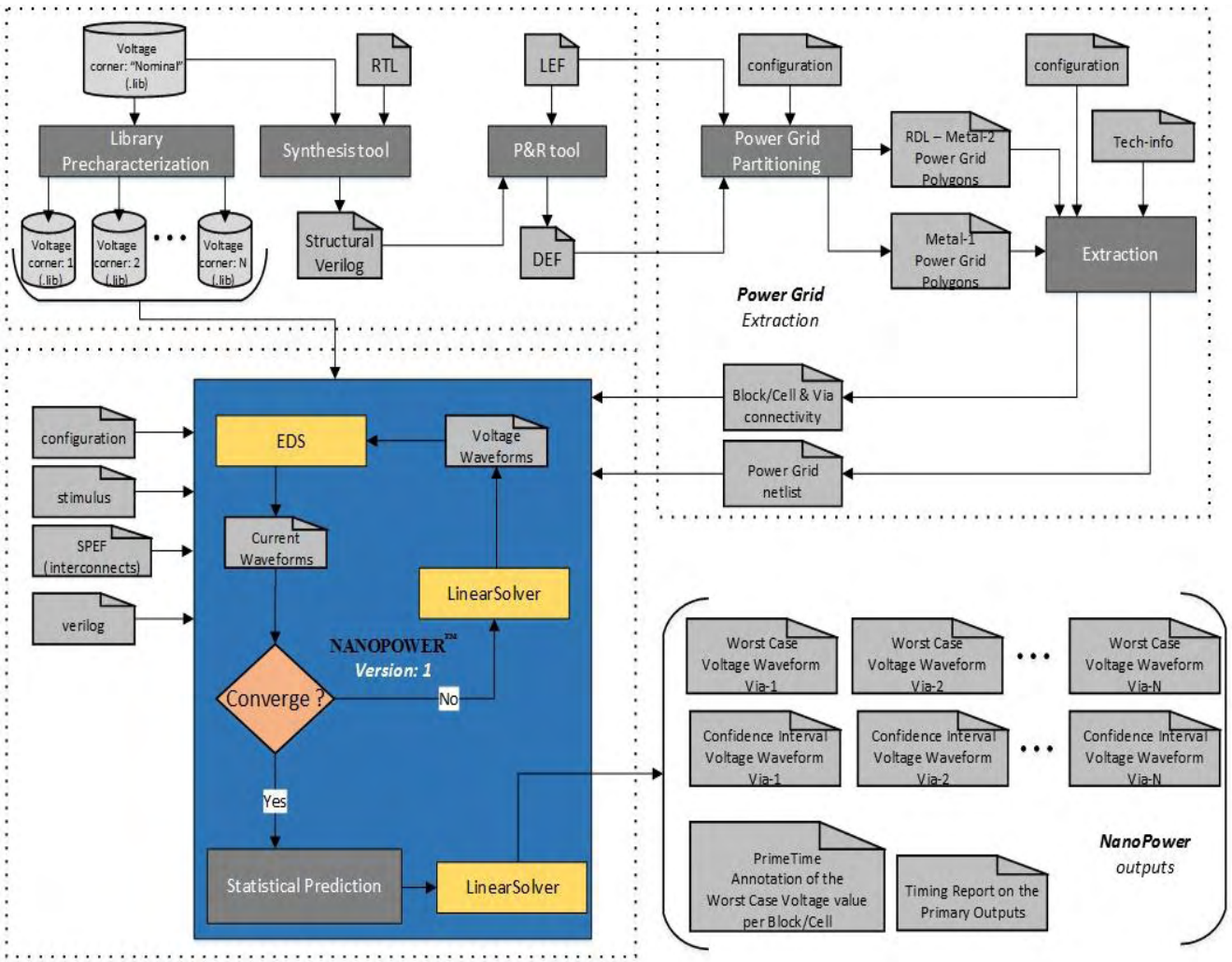


Figure 9 Proposed methodology – version: 1 (NANOPOWER™ tool).

The next generation of the tool (**Figure 10**) has a slightly different approach. The main modules of the tool are exactly the same as in version 1 but in this approach (version 2) the Statistical Prediction Engine takes as input the voltage waveforms resulted from LinearSolver, after the final iteration of the methodology (both current and voltage waveforms have converged within a pre-specified error).

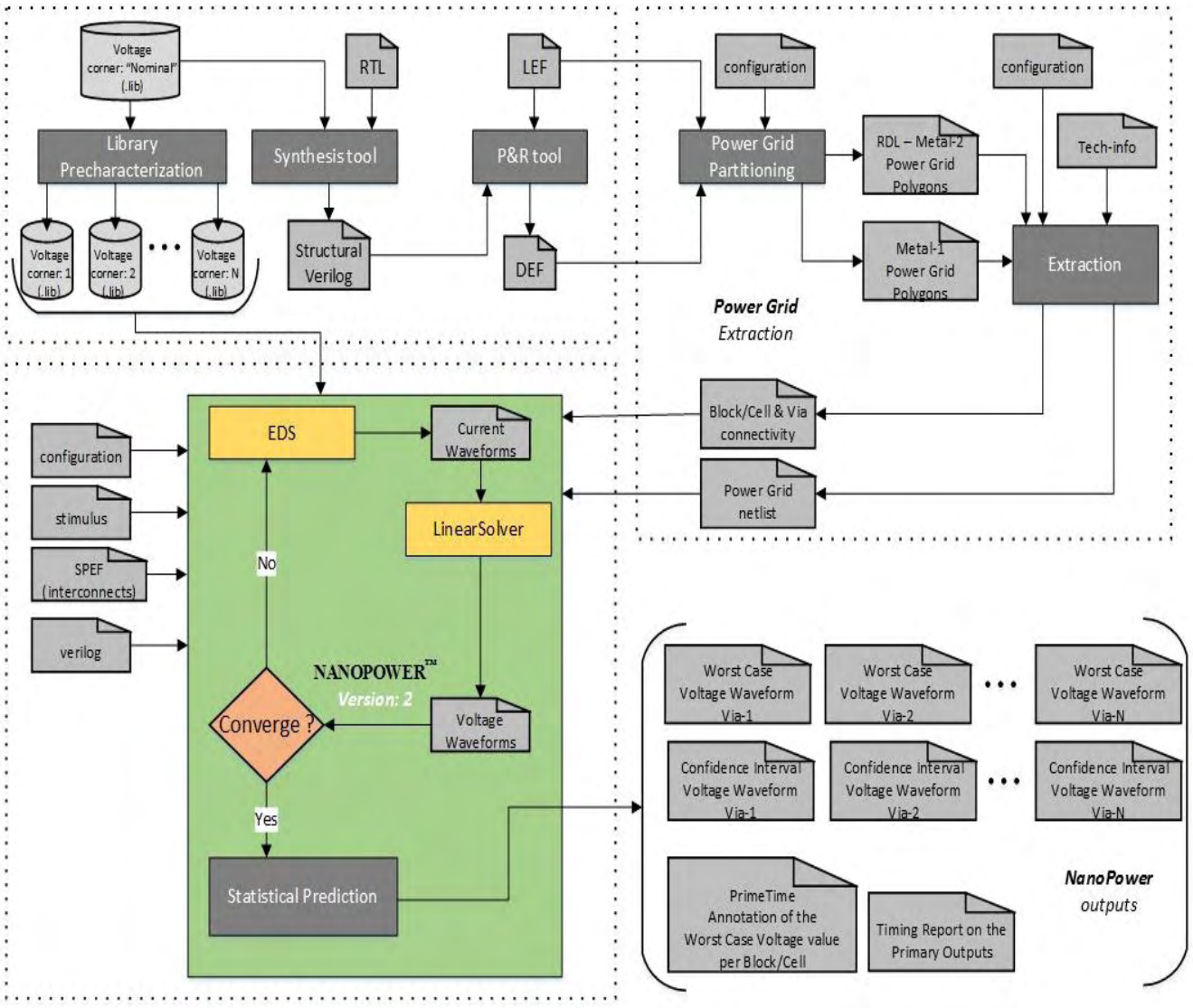


Figure 10 Proposed methodology – version: 2 (NANOPOWER™ tool).

Both versions of the proposed methodology report an accurate, tight upper bound of the worst case voltage waveform for each via (tap-point) of the power/ground supply network.

5.4.1 Comparing the Proposed Methodologies

Although both versions of the tool deliver accurate results they have advantages and disadvantages. The following list describes the main advantages of each version of the proposed methodology.

- NANOPOWER™ version 1:
  - The current waveforms (total power) of the IC do not change and thus it is easier and much faster to make fixes on a power/ground supply network and run only the last step of the methodology (LinearSolver) to observe the results.

- NANOPOWER™ version 2:
  - More accurate on the final results since there is always the possibility that two or more digital blocks of the design are mutually exclusive. In this case, version 1 would report the worst case voltage waveforms resulted from the simulation of the power/ground supply network applying on it, simultaneously the current waveforms drawn by mutually exclusive digital blocks (**Figure 11**), which is not a real scenario. The resulted waveforms would be pessimistic.
  - It is faster since LinearSolver, which is the heaviest module of the tool in terms of runtime will run one more time.

As it easily deduced the first approach (version: 1) is the appropriate one when the designer discovers a Power Integrity issue coming from a bug on the power/supply network design, since the fix and the corresponding verification will be much faster. When it comes to accuracy it is obvious that the second approach (version: 2) is more accurate and faster. Since both versions report the worst case voltage over each cell of the IC in order to be used in Timing closure, accuracy is of the essence and thus the second approach (NANOPOWER™ version 2) prevails.

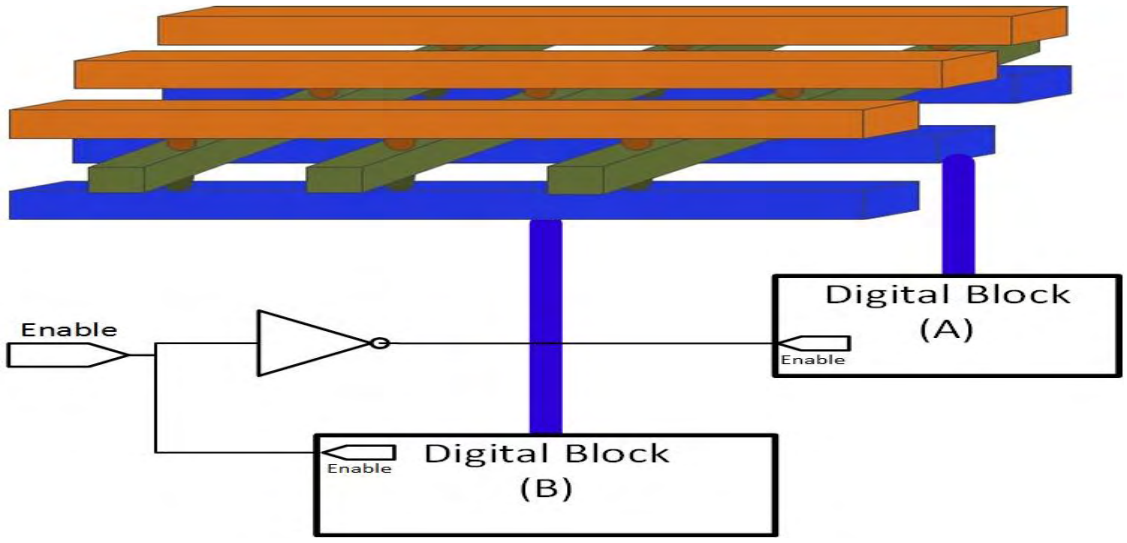


Figure 11 Neighboring mutually exclusive Blocks

5.4.2 Real Worst Case Voltage-Drop

Existing methodologies for power integrity analysis use current waveforms for the simulation of the power supply network, the waveforms generated at the power/ground pins of the cells and macros of the design, which result from the activity of these elements during IC operation. This approach assumes erroneously that the voltage level over the power/ground pins of each cell during their activity is the nominal (ideal) voltage level and thus the waveform signatures generated, are far away from the real ones. This issue has already been described in sub-section 5.2.2.3. Existing logic simulators are not able to incorporate several libraries in different voltage corners. The proposed hybrid approach for power integrity verification has two major advantages, the one is the accurate calculation of the voltages and currents at the power grid nodes and the second one is the accurate calculation of the logic events generated during the logic simulation.

During NANOPOWER™ operation, each vector from a stimulus file (VCD file) is used for the generation of the initial logic-events at the primary inputs of the design (SAIF file can also be used). These time dependent logic events are propagated through the design, using an event driven logic simulator, causing activity on the cells which in turn, draw current from the power supply network, through their power pins, flowing to their output pins. For the delay calculation of each logic event generated at the output port of each cell/block, the tool is aware of the voltage over the VDD pin of each cell/block and uses the data from the corresponding library (liberty file). Most of the times, in order to calculate the delays and currents, the simulator has to interpolate between two libraries precharacterized in different voltage levels. Through this type of simulation the event driven simulator is able to capture (handle) glitches generated at the cells during IC operation. The tool calculates the transient current waveforms ('signatures'), which have resulted from the switching activity, for each via connecting the lower metal stripes (Metal1 commonly), where the power pins of the cells are connected with the rest of the supply network. For this calculation precharacterized libraries, in several different voltage levels are used. These current signatures are then attached to the corresponding spice nodes in the form of independent current sources. An internal circuit simulator, for passive netlists, is used in order to simulate the final circuit produced after modeling the power supply network as passive RC/RLC netlist using HELIC's RaptorX™. The resulted voltages are fed back to the NANOPOWER™ simulation engine to enhance the accuracy of the result. This implies an iterative process. Usually, 3-5 iterations between the two simulators (event driven simulator and circuit simulator) are enough to converge to within 2-3% of SPICE results (generated with benchmarks against transistor-level SPICE on designs with a few thousand gates) as depicted in **Figure 12** of the initial input vectors. After 3-4 iterations, the user obtains the exact voltage waveforms, over each via of the IC's supply net, resulted by the simulation. Despite the accurate results regarding the voltage waveforms over the cells, after simulating a small subset of the total amount of the input vectors, the tool has to overcome the bottleneck of finding those input vectors that excite the worst case voltage drops on the design. In order to surpass this bottleneck NANOPOWER™ makes use of a powerful statistical prediction engine based on a solid mathematical foundation, which takes as input voltage drop waveforms coming from the simulation of the design using a small number of input vectors and predicts the worst case voltage drop waveforms over all possible input vectors.

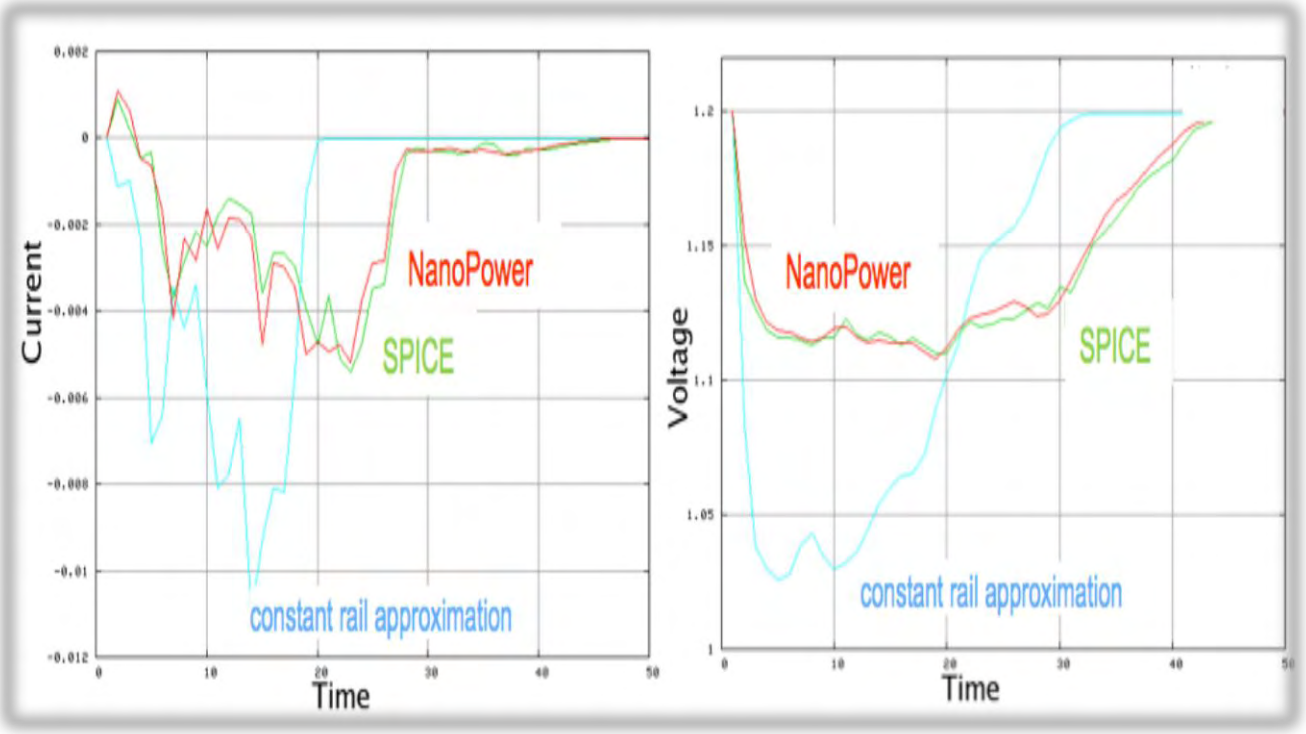


Figure 12 NANOPOWER™ Voltage and Current waveforms vs corresponding waveforms from Spice simulator vs Constant Rail approximation (1.0V nominal Power Supply level).

5.4.2.1 Experimental Setup and Results

For the validation of the proposed methodology (NANOPOWER™ tool), two large scale industrial designs and several ISCAS benchmarks are used. The ISCAS were synthesized using NanGate-45nm Process Design Kit using Synopsys Design Compiler. To be able to test the accuracy of the methodology Voltage-drop and ground bounce had to be present in the design. For this reason both Power and Ground Supply Networks were changed by reducing the widths of several metal stripes for both Networks in a pre-specified area for each design. Moreover in the large designs current sources were connected to the power and ground networks to emulate the power consumption of a large digital block. **Table 1** reports the power consumption calculated by the tool along with the worst case voltage reported both in Power Supply Network (column: *Voltage-drop*) and Ground Network (column: *Ground-bounce*). To stimulate the designs random input vectors were used. The Power and Ground Supply networks were extracted using HELIC's RaptorX™. The color-maps of the runs depict the worst-case voltages in areas of the designs where the metal stripes were intentionally changed in order to be more resistive and also areas where extra current sources were connected proving that the proposed methodology is extremely accurate and points out in detail the areas along with the included cells, blocks and Power Supply Sub-networks that suffer from voltage-drop.

Table 1 NANOPOWER™ - Power Integrity simulation results.

Design	Power Consumption (mW)			Voltage-drop (Volt)	Ground-bounce (Volt)
	Dynamic	Leakage	Total		
H264	36.8510	17.325191	54.176191	0.93268	0.157130
JPEG	9.40359	3.8508950	13.254480	0.89490	0.227860
S38584	5.47201	2.1566700	7.628690	0.89856	0.372541
C7552	3.99410	0.3000300	4.294130	0.88940	0.281135
C6288	11.6315	0.1734370	11.80500	0.87890	0.365480
Wallace	1.08379	0.0906410	1.174431	0.84789	0.134587
C5315	2.45365	0.2070360	2.660690	0.90510	0.236550
C3540	2.32413	0.1440860	2.468220	0.91210	0.056630
C2670	1.61037	0.1101790	1.720550	0.95820	0.033590
C1908	1.79948	0.0672401	1.86672	0.96110	0.012875
C1355	0.49382	0.0415404	0.53537	0.96418	0.014184

The proposed methodology (NANOPOWER™ tool) reaches Spice accuracy extremely fast and is able to simulate industrial size ICs. The methodology is fully parallelizable both in the flow level but also in almost all its individual steps (e.g. the logic simulation). **Figure 13** depicts the tool's runtime for 1M gates and 10M gates.

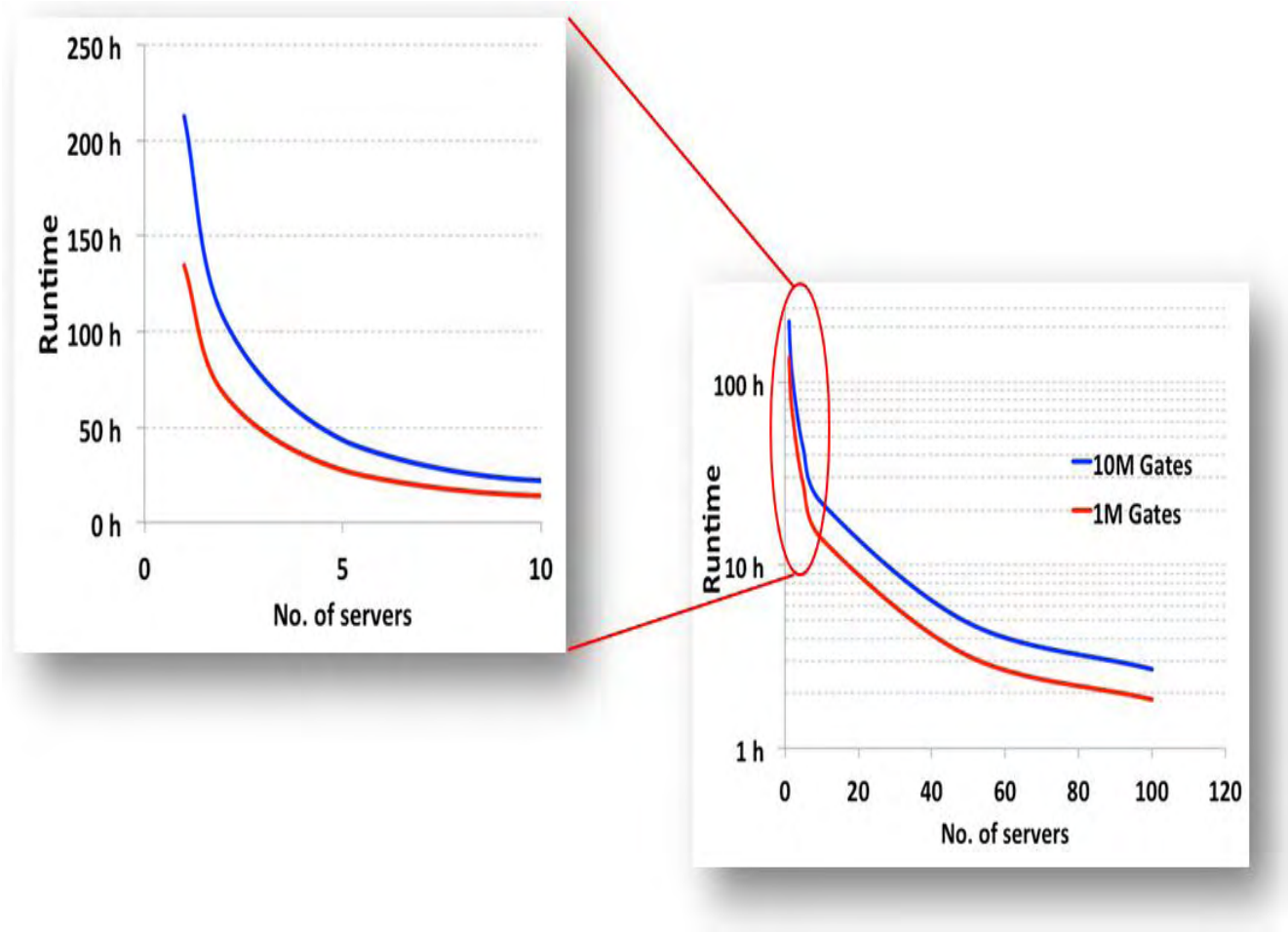


Figure 13 NANOPOWER™ runtime graph for 1 Million gates and 10 Million gates.

The LinearSolver module is the bottleneck for the runtime of NANOPOWER™, a fact that becomes clearer when observing the profiling results of the tool for two industrial test-cases, a JPEG core and an H264 core as depicted in **Figure 14** and **Figure 15** respectively. The larger and more complex o power/ground supply network is, the longest takes for the LinearSolver to simulate it, since there is not enough space for parallelization for these algorithms.

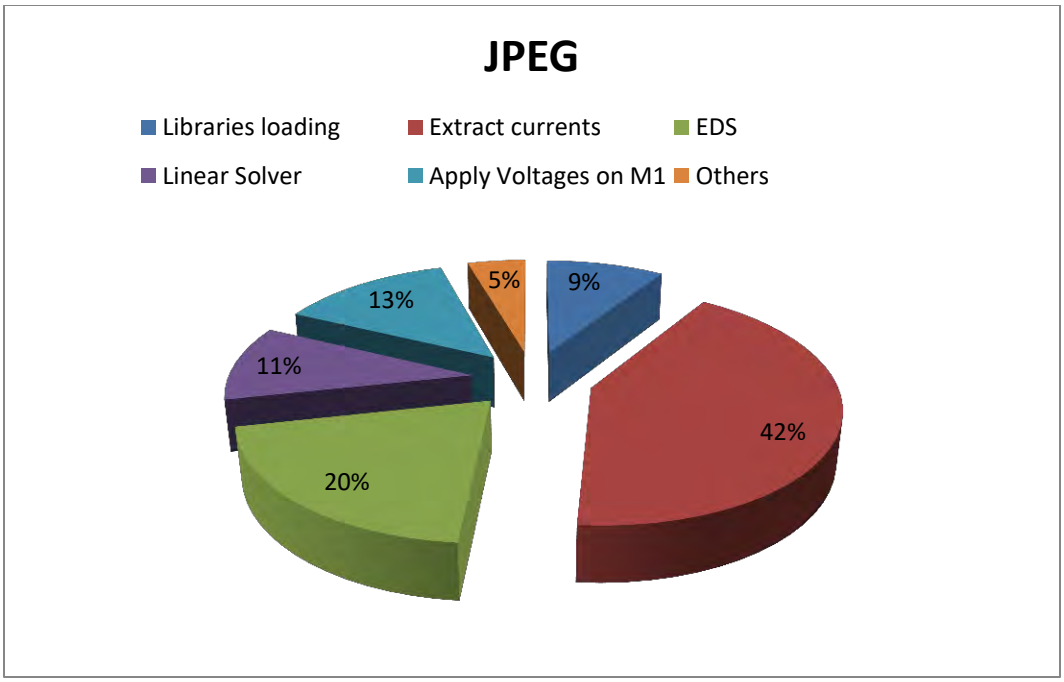


Figure 14 NANOPOWER™ profiling for JPEG.

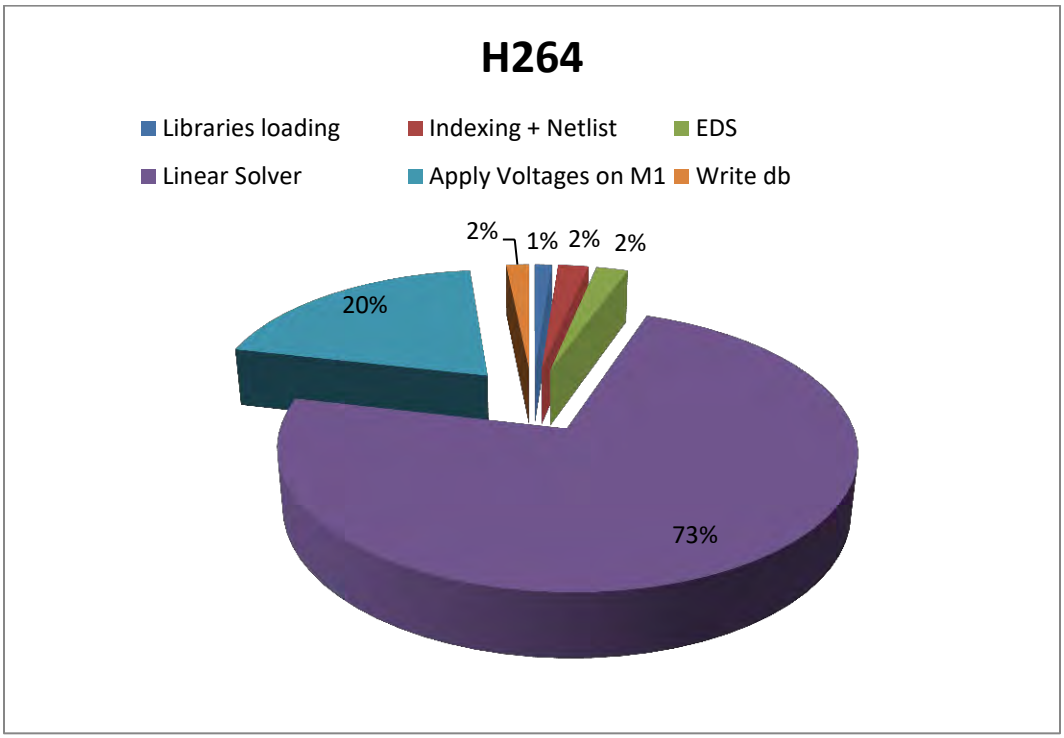


Figure 15 NANOPOWER™ profiling for H264.

5.4.3 Fast transform-based preconditioners for large-scale power grid analysis on massively parallel architectures

The main problem of iterative methods is their unpredictable rate of convergence which depends greatly on the properties (specifically the condition number) of the system matrix (**Figure 16** vs **Figure 17**). A preconditioning mechanism, which transforms the linear system into one with more favorable properties, is essential to guarantee fast and robust convergence. However, the ideal preconditioner (one that approximates the system matrix well and is inexpensive to construct and apply) differs according to each particular problem and each different type of system matrix. That is why iterative methods have not reached the maturity of direct methods and have not yet gained widespread acceptance in linear circuit simulation. Although general-purpose preconditioners (such as incomplete factorizations or sparse approximate inverses) have been developed, they are not tuned to any particular simulation problems and cannot improve convergence by as much as specially-tailored preconditioners.

Another aspect of circuit simulation that has become very important recently is to uncover hidden opportunities for parallelism in its intermediate steps. This is essential for harnessing the potential of contemporary parallel architectures, such as multi-core processors and graphics processing units (GPUs). GPUs, in particular, are massively parallel architectures whose computational power is about 1580 GFlops/s, greater by an order of magnitude than that of multi-core processors, and as a result they appear as a platform of choice for the efficient execution of computationally-intensive tasks. However, there has been little systematic research for the development of parallel simulation algorithms, and more specifically algorithms for power grid analysis that can be mapped onto massively parallel architectures like GPUs. This can be attributed in part to the difficulty in parallelization of direct linear solution methods that have been mostly employed thus far.

On the contrary, Krylov-subspace iterative methods offer ample possibilities for parallelism that have been explored sufficiently well. However, the construction and application of the preconditioner is a very delicate part of parallelizing an iterative method because it is completely application-dependent (and traditional general-purpose preconditioners have very little room for parallelism). The design and implementation of a parallel preconditioning mechanism that can be used in conjunction with a Krylov-subspace method, such as Preconditioned Conjugate Gradients (PCG), for efficient DC or transient analysis of power distribution networks has been developed.

A proposed preconditioner is constructed in a way to approximate very closely the power grid under analysis, so that the total number of iterations of the iterative method is reduced to a great extent. At the same time, its specialized structure allows applying a Fast Transform-based solver that utilizes Fast Fourier Transform (FFT) for the solution of the necessary preconditioning step. The main characteristic of the application of a Fast Transform is the near-optimal operation complexity, as well as its inherent parallelism and low memory requirements, compared to a generic solver for linear systems. As a result, massively parallel architectures such as GPUs can be used to accelerate the simulation algorithm, while at the same time the application's memory demands, and render feasible the analysis of large power grids on such architectures. Experimental results show that our algorithm implemented on a GPU and applied on a 2.6M-node industrial power grid achieves a speedup of 214.3X over CHOLMOD [3] (a state-of-the-art direct linear solver) and 138.7X over a parallel version of PCG with incomplete Cholesky preconditioner implemented on a multi-core CPU. The efficiency of the proposed preconditioner is more profound as the design size increase where a speed-up of 1610.5X and 438X over CHOLMOD and the parallel version PCG is achieved for a 3.1M-node synthetic design, when GPUs are

utilized. At the same time, its matrix-less formulation allows for reducing the memory footprint by up to 33% compared to the memory requirements of the best available iterative solver.

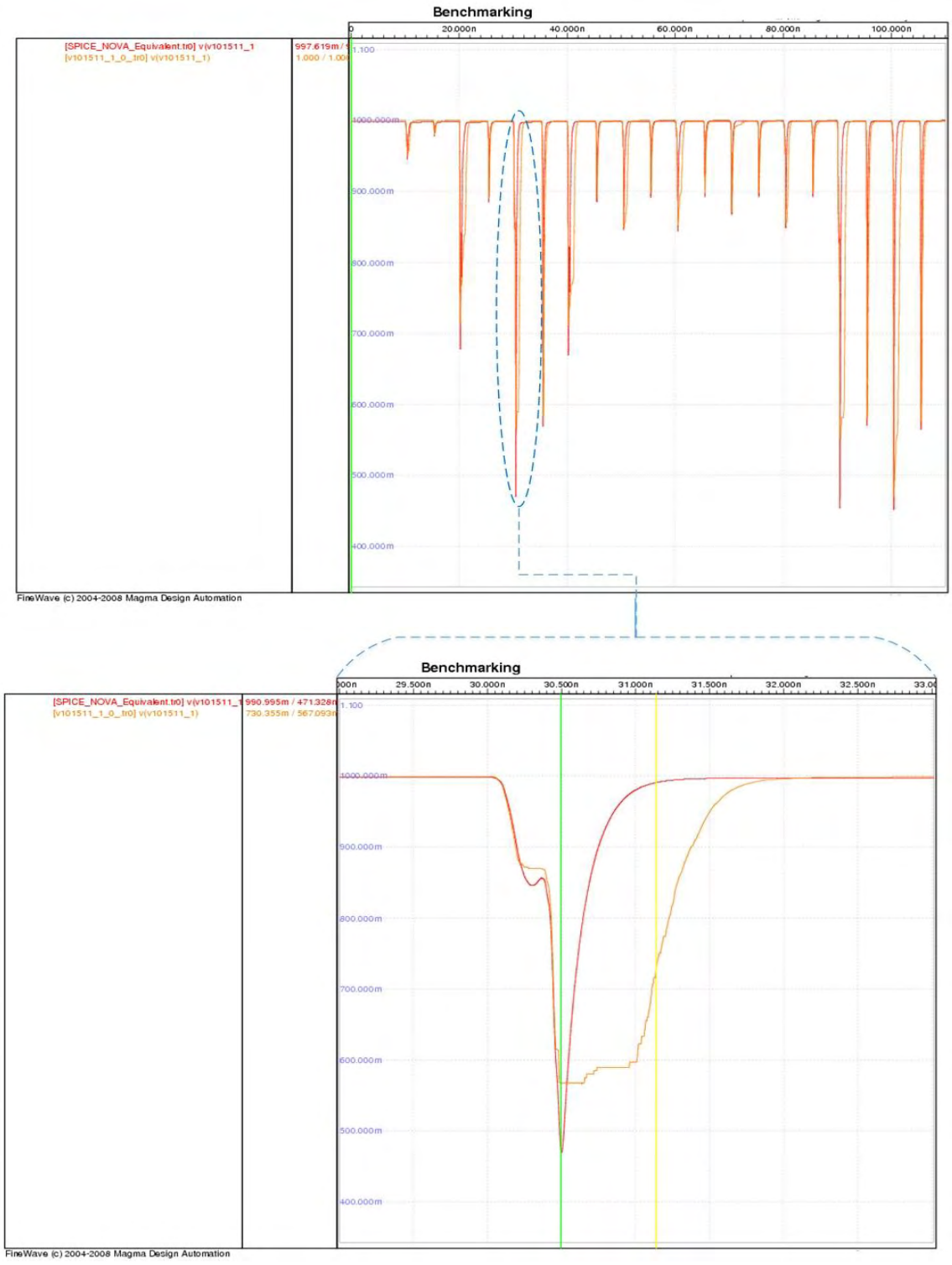


Figure 16 H264: HSPICE vs NANOPOWER™ using Linear Solver Tolerance 1e-02 and maxit200.

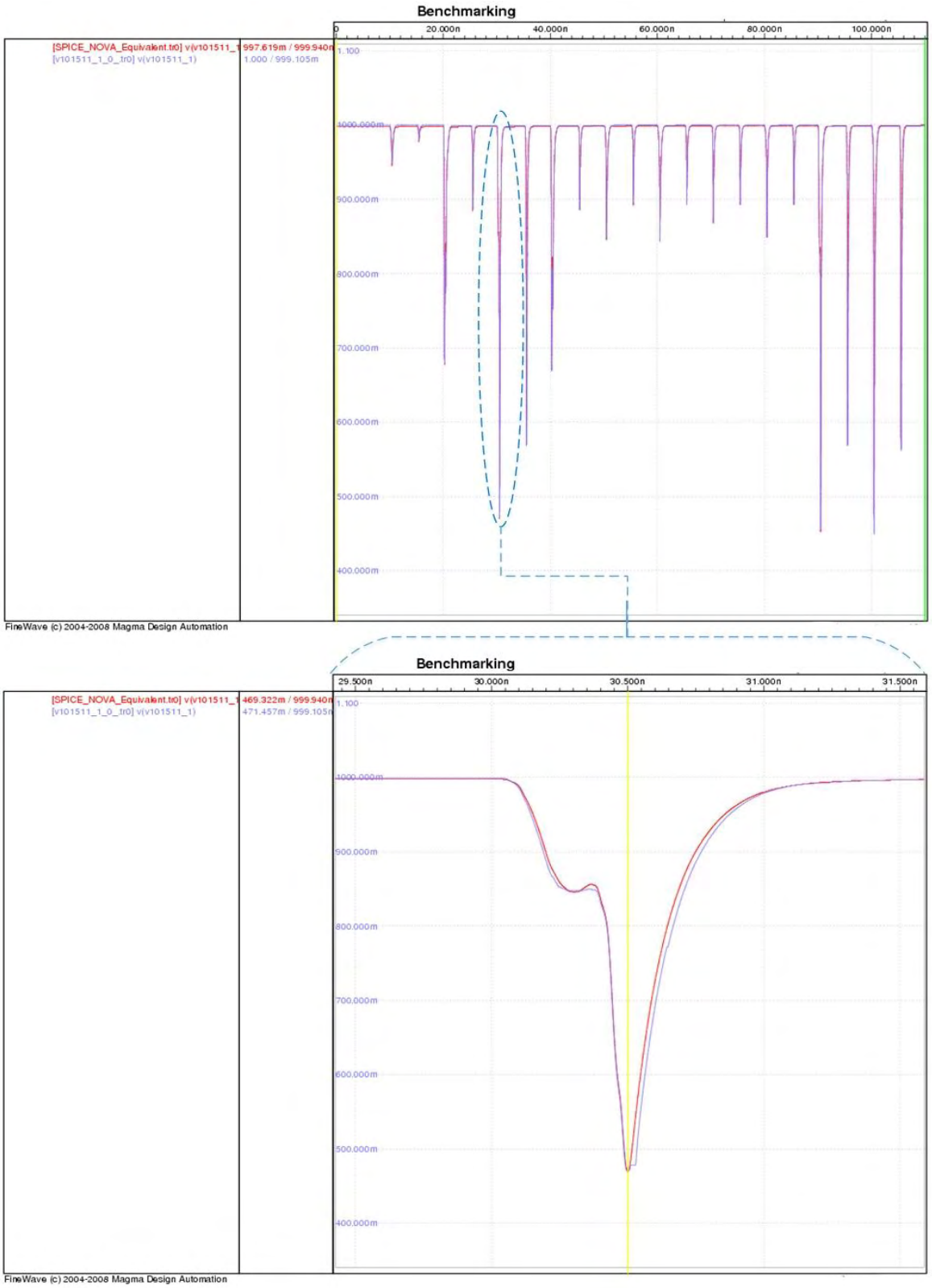


Figure 17 H264: HSPICE vs NANOPOWER™ using Linear Solver Tolerance 5e-03 and maxit500.

5.4.3.1 Statistical Prediction Engine

One of the major bottlenecks for the prediction of the worst case IR-drops on the power delivery network, is to identify the specific vector sequence that trigger the placed logic-gates (standard cells) in the layout of the IC to draw the amount of current which in turn lead force the power supply network to suffer the worst-case voltage drops. The determination of those vector pairs is a NP-complete problem – i.e. cannot be solved analytically. The independent approaches for the estimation of the maximum voltages which have appeared over the years were mostly heuristic or over-simplified and could not provide the accuracy needed for the design of deep-submicron ICs. NANOPOWER™ makes use of a statistical prediction engine based on breakthrough extensions of Extreme Value Theory (EVT) tailored to provide a tight estimate for the true worst-case voltage drop at each point of the power supply network, using the current and voltage waveforms generated by the NANOPOWER™ simulator. The application of EVT to this problem eliminates the need to identify those vector pairs that generate the worst-case voltage drop, since the last can be estimated from the simulation results of a finite random set of vector pairs, not necessarily containing the worst-case pair.

In order to implement our methodology we have to acquire an initial sample $S = \{y_1, y_2, \dots, y_l\}$ of mN-dimensional current waveforms (henceforth referred to as the "sample space") by simulating the place and routed IC for l random vector pairs $\{b_p, b_n\}$. This multivariate sample is made up of an assortment of mN univariate samples $S = \{y_{i,1}, y_{i,2}, \dots, y_{i,l}\}, \quad i = 1,2, \dots, mN$, each one representing the current waveforms observed at each Via of the power supply network, at a particular time instance and constitutes the result of the IC simulation, concerning the specific Via, for the l random vector pairs $\{b_p, b_n\}$. In each univariate sample S_i we can estimate the expected maximum $\omega\,(y_i)$ of the random variable y_i (instant voltage) sampled by S_i as follows:

$$(5.6)\quad \widehat{\omega}(y_i) = \widehat{\mu}_l + \frac{\widehat{\sigma}_l}{1+r\sqrt{\pi \log r}\left(\mathrm{erf}\left(\sqrt{\log r}\right)-1\right)}$$

Where $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$ is the "error function" and $\widehat{\mu}_l$, $\widehat{\sigma}_l$ are estimates of the location-scale parameters of the asymptotic extreme value distribution (not related to the corresponding parameters of the normal distribution), which are obtained by the method of matching the first and second moments (i.e. mean and standard deviation) of the sample Z_i with those of the extreme value distribution, by which we have:

$$(5.7a)\quad \widehat{\sigma}_l = \left(\sqrt{6}/\pi\right) std(Z_i)$$

$$(5.7b)\quad \widehat{\mu}_l = mean(Z_i) - \gamma\widehat{\sigma}_l$$

where $\gamma \approx 0.5772$ ... is the "Euler gamma" constant. The units of S are completely random and do not have to be close to the (unknown) worst-case vectors, since the maximum $\omega\,(y_i)$ for each $i = 1,2, \dots, mN$ is actually estimated via the parameters of the extreme value distribution which is followed asymptotically (i.e. for large enough r) by the sample of maxima Z_i. In order to estimate the worst case currents we have to determine the excitation space D starting from the initial sample space S consisting of all the current waveforms that resulted from the IC simulations using the initial l random vector pairs$\{b_p, b_n\}$. The sample space $S = \{y_{i,1}, y_{i,2}, \dots, y_{i,l}\}$ has a set of maximal points of its own, which will be scaled *down* in each individual coordinate $i = 1,2, \dots, mN$ (**Figure 18**) with respect to the maximal subset of the excitation space D (since there will always be points $y \in D$ lying outside the outermost boundary of S). A reasonable approximation for this down-scaling of the

maximal subset as a *whole* in each $i = 1,2,\dots,mN$ is $\omega(y_i) - max\{y_{i,1}, y_{i,2}, \dots, y_{i,l}\}$, where $max\{y_{i,1}, y_{i,2}, \dots, y_{i,l}\}$ is the maximum value of each univariate sample S_i (i.e. the maximum of the sample space *S* in each coordinate axis). Writing this succinctly in vector form for all $i = 1,2,\dots,mN$ as:

$$(5.8)\quad \boldsymbol{d} \equiv \omega(y_i) - max\{y_1, y_2, \dots, y_l\}$$

[where the max operator in interpreted component-wise] we have a difference vector by which we can shift the maximal subset of *S* in order to move it to the expected location of the maximal subset of *D* in $\mathbb{R}^{mN}$. It must, of course, be mentioned that the maximal subset of *D* will have much different structure and include many more points than the maximal subset of *S*, but the maximum value of a linear function is fairly insensitive to the local structure of the maximal subset and instead depends predominantly on its global position in $\mathbb{R}^{mN}$ (**Figure 19**). In order, finally, to compute the maximal points of the space *S* consisting of *l* points, we have to compare each point to all others (to determine whether a specific point is *not* dominated by *any* others in *all* components, according to Definition 1), which leads to a total of l^2 comparisons. It can be shown however, that the necessary comparisons can be reduced to at most $O(l(\log_2 l)^{mN-2})$, where mN is the dimension of the space and its constituent points.

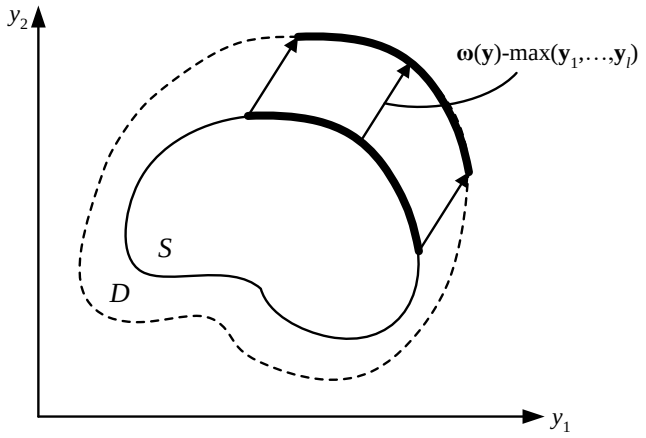


Figure 18 Sample space S and shift of its maximal points towards the expected position of the maximal points of the excitation space D.

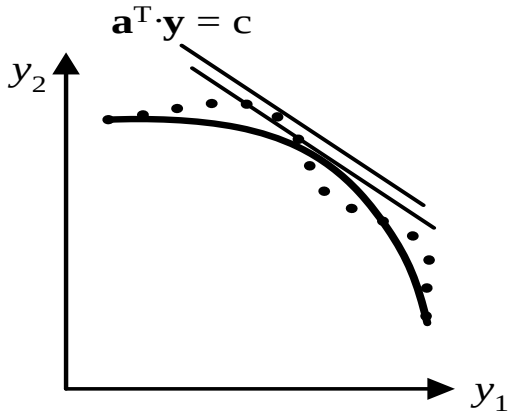


Figure 19 Insensitivity of the maximum of a linear function to the local structure of the subset of maximal points.

The process behind the Statistical Prediction Engine can be described by the following 8 steps:

1. *Generate a total of $l = 2500$ random pairs of binary vectors $\{b_p, b_n\}$ for the circuit under consideration.*
2. *Simulate the circuit under all generated pairs and record the discretized current waveforms in each sink $j = 1,2,\dots,mN$ and for each time instant $t = kh,\ k = 1,2,\dots,N$ within an interval of interest (e.g. a clock period). The recorded data $S_i = \{y_{i,1}, y_{i,2}, \dots, y_{i,l}\},\ i = 1,2,\dots,mN$, taken jointly as mN - dimensional vectors will constitute the sample space $S = \{y_1, y_2, \dots, y_l\}$.*
3. *Arrange each univariate sample $S_i\ i = 1,2,\dots,mN$ into $\frac{l}{r} = 100$ sub-samples of size $r = 25$. Here the size r only needs to be adequate so that the sample of the maxima units from the sub-samples follows an asymptotic extreme value distribution. We have found by experimentation that $r = 25$ is a fair value. The number $\frac{l}{r} = 100$ of sub-samples (leading to a total of $l = 2500$ units) yields estimates with relative estimation error (i.e. quotient of confidence interval to estimate) of about 5% – at a confidence level 95%*
4. *For each $i = 1,2,\dots,mN$ construct the sample Z_i of the maxima units from the l/r sub-samples of S_i.*
5. *For each $i = 1,2,\dots,mN$ calculate the estimates $\hat{\mu}_i$, $\hat{\sigma}_i$ of the extreme value distribution parameters from (2a, 2b), and the estimate $\hat{\omega}(y_i)$ of the expected maximum $\omega\ (y_i)$ from (1).*
6. *Determine the maxima $max\{y_{i,1}, y_{i,2}, \dots, y_{i,l}\}$ of all univariate samples $S_i,\ i = 1,2,\dots,mN$, and in conjunction with the estimates $\hat{\omega}(y_i),\ i = 1,2,\dots,mN$ for the expected maxima, compute the mN-dimensional difference vector **d** from (3).*
7. *Locate the maximal points among the $l = 2500$ points of the sample space S. As already mentioned, this step has complexity of $\mathrm{O}((log_2\, l)^{mN-2})$ comparisons.*
8. *Shift the maximal points of the sample space S by the computed difference vector **d**. This step is performed by plain component-wise addition of the vector **d** to the maximal points of S, and is a trivial one.*

The extension of the Statistical Prediction Engine for to cover the second version of the tool is straight forward since the sample space in this case is the total number of voltage waveforms on each Via of the design. Performing EVT using all voltage waveforms of each Via (**Figure 20** colored waveforms between bold blue

and bold red waveforms, which are the envelopes) provides the worst case voltage waveform (**Figure 20** bold green waveform) for that Via.

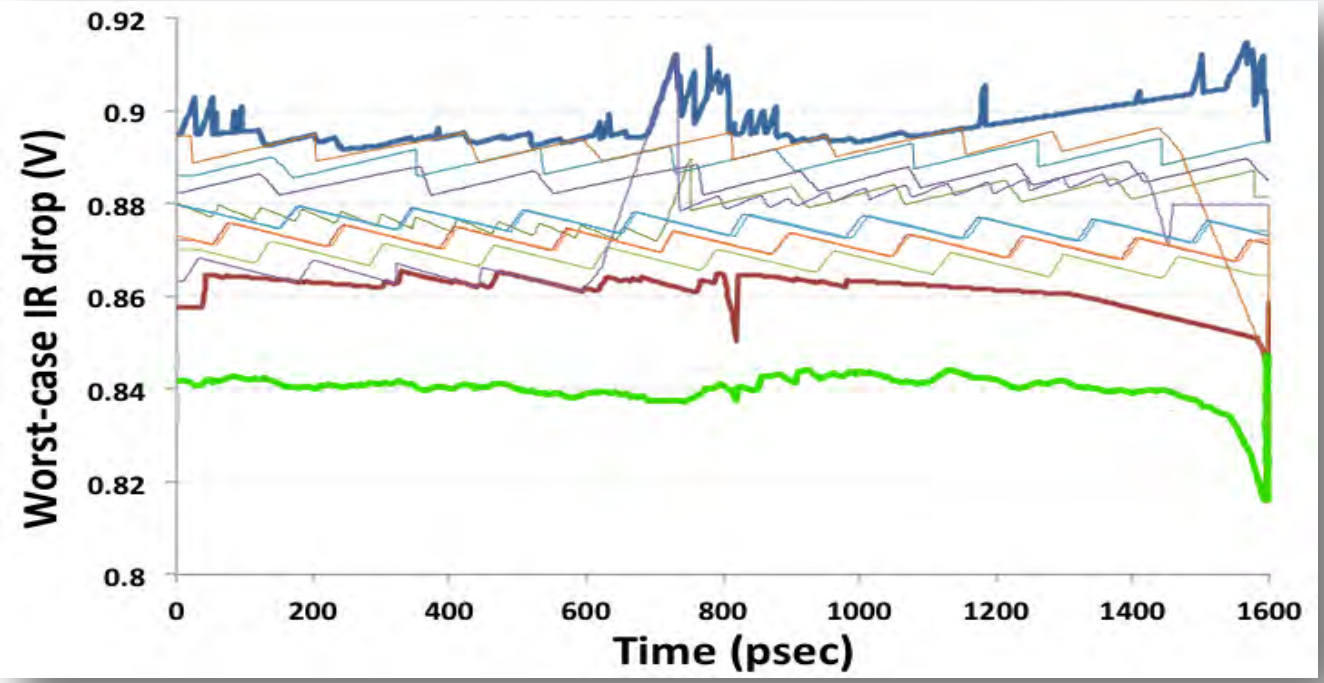


Figure 20 Voltage waveforms from simulation (above) and the corresponding Worst-Case voltage waveform (below-green) estimated by the Statistical Prediction engine (EVT).

5.4.3.2 Confidence Interval of Statistical Prediction Engine

Since the Statistical Prediction Engine uses EVT, which is a theory based on a statistical model, there is clause characterizing the quality of the results, this clause is the Confidence Interval. Each waveform that results from the Statistical Prediction Engine is accompanied by a confidence level. A confidence interval (corresponding to a confidence level of $(1-\delta)\times 100\%$ has also been constructed for the estimate (1), and is given by:

$$(5.9)\quad |\widehat{\omega}-\omega| \le \frac{Z_{\delta/2}}{\sqrt{m/l}}\frac{\hat{b}_n\sqrt{6}}{\pi}\cdot\sqrt{(\gamma-1)^2+\frac{\pi^2}{6}+\frac{2(1-\gamma)}{1+l\sqrt{\pi\log l}\,(erf(\sqrt{\log l})-1)}+\frac{1}{\left(1+l\sqrt{\pi\log l}(erf(\sqrt{\log l})-1)\right)^2}}$$

Where $Z_{\delta/2}$ is the $\delta/2$ quantile point of the standard normal distribution and $\gamma \approx 0.5772\ldots$ is the "Euler gamma" constant. A waveform with Confidence level of 99% means, there is 99% probability the true worst case voltage for each via in the design and for each time value to be inside the confidence interval, estimated by the engine, relatively to the estimated worst case voltage. As a conclusion of the above statements, each via of the design can be characterized by three basic waveforms resulted from the Statistical Prediction Engine. The basic waveform is the worst case voltage waveform which gives us the worst case voltage values for each time value in one period. The other two waveforms are used due to the nature of the non 100% accuracy of the

results and are the confidence interval waveforms. The confidence interval waveforms (**Figure 21** bold blue and bold orange waveforms) set the space of the voltage values for each time value in a period where the true worst case voltage will be. In this way, the confidence interval waveforms give the lower and upper bounds where the true worst case voltage is guaranteed by the engine to appear during simulation.

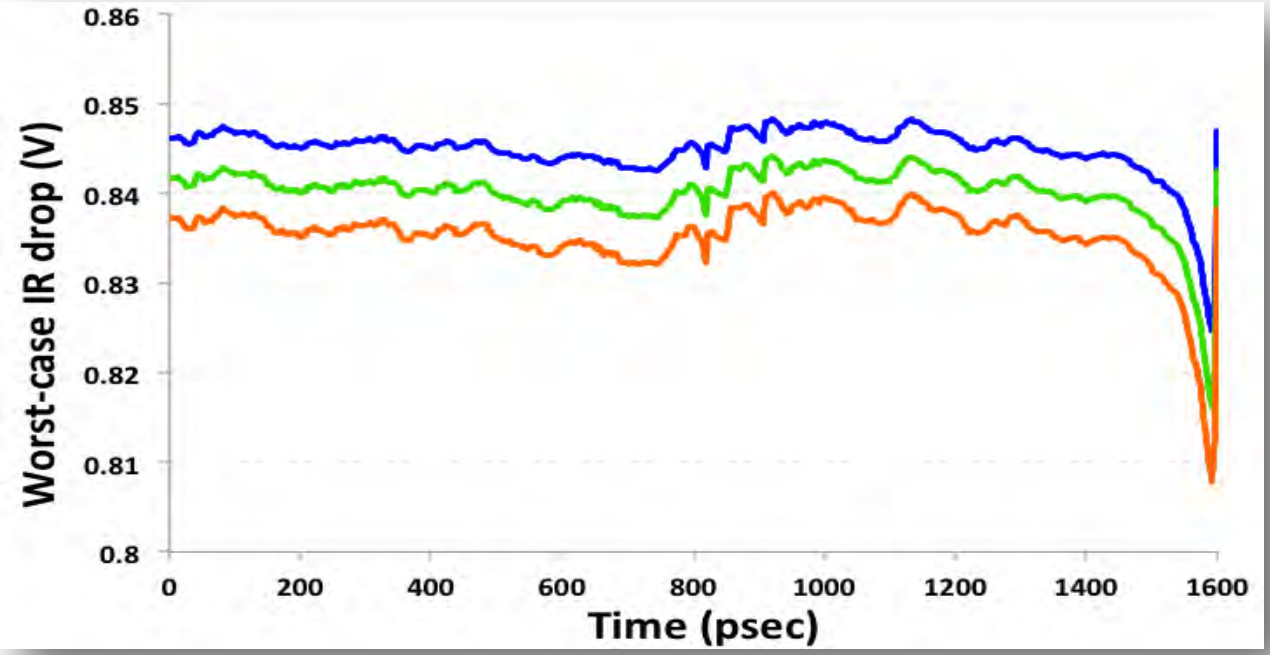


Figure 21 Confidence Interval waveforms (orange-blue) and the corresponding Worst-Case voltage waveform (green). In the specific plot the confidence has been set to 99%.

5.4.3.3 Sample Space Generation in Multi-Modal Designs

Modern ICs function in multiple different modes of operation. Each mode of operation produces different switching activity on the IC, which means that each input vector triggers a different combination of cells or blocks in the IC located also in different areas. In **Figure 22**, the four lobes of the distribution graph depict the current consumption of an industrial IC in four different modes of operation. Each instance depending on its type and its input ports switching, consumes different amount of power and of course, as described in the sub-section 5.2.2.3, if the voltage on its power supply pin is non ideal during switching, which is the common case, the current waveform on its power supply pin will also differ from a current waveform produced considering ideal power supply voltage on the power supply pin. If one considers the fact that the voltage-drop on an area of the IC may result from the switching activity of another part, possibly neighboring block on the IC, it is easy to deduce that it is rather unfeasible, to either find out the exact combination of input vectors that will produce the worst case voltage drop for each cell of the design or even to find out the worst case voltage drop on the IC for all possible input vectors. This also means that the attempt to find the worst case voltage on a tap-point (Via connecting Metal-1 with the rest of the power grid) of the power grid, by summing up the individual current waveforms from the power supply pins of the cells from both sides of this tap-point, and then try to find the

worst case voltage-drop at this tap-point, would not be feasible. At this point there are several arising issues for clarification on the estimation of the worst case voltage-drop. The basic issue is the proper choice of the initial input vectors used for the simulation of the design under test, which is the part of the power integrity process which generates the voltage waveforms on the power supply network used as the sample-inputs of the statistical prediction engine, which will estimate the worst case voltage waveform on every tap point of the design.

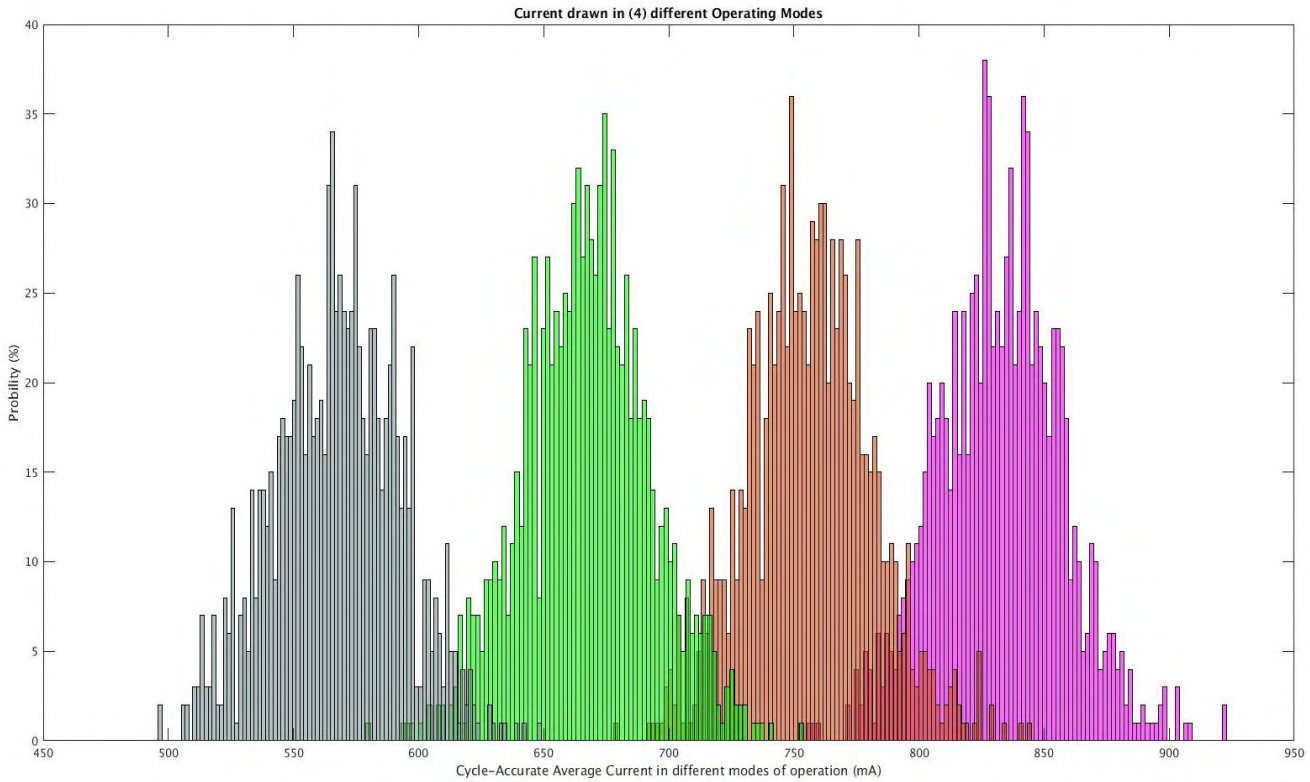


Figure 22 Distribution of Currents drawn in different modes of operation.

For a multi modal design, the set of all possible different input vectors that can be used for stimulating the design is separated into several subsets. Each of these input vector subsets can be used in the design, depending on the mode of operation the designer wants to test the IC for power integrity.

Seeking for a set of input vectors to drive the power integrity analysis, leads us to search for a set of vectors that will generate the proper samples for the statistical prediction engine. This means that the input samples of the statistical prediction engine, in our case the voltage waveforms, should not be biased and not include outliers. The samples of the statistical prediction engine may be considered to be biased in case they result from modes of operation that report samples with values close to each other having a mean value far from the real one and include outliers when the samples include data that are far away from the mean values. Since the statistical analysis estimates the worst possible voltages that may occur at each tap-point of the power supply network, the samples used should not be far from the real worst case voltage waveforms.

In order to find a valid set of input vectors one should first answer the question “which are the reasons behind the voltage drop effect” an answer that now becomes of high importance. This question has already been answered briefly in sub-section 5.2.2.3. The main reasons behind voltage-drop effect are the large amount of current simultaneously drawn by the cells of the design during simulation and a badly designed power supply network. During power integrity analysis the power supply network of the design under test, has already been

extracted and modeled, so it is considered to be something fixed during this process. In other words if one formulates the problem with an equation the only unknown variable which has to be found, is the exact signature of the current waveforms on the power supply network of the IC during the simulation. As we have already seen, the proposed methodology implemented with NANOPOWER™ tool gives enough accuracy for the output voltage waveforms. Since the only variable, from users' perspective, that impacts the voltage waveforms and consequently the sample space of the statistical prediction engine is the currents and since the samples space should not contain outliers, the valid currents to be used are those generated in the mode of operation with the highest power consumption. As a result the input vectors used for the power integrity analysis should be the input vectors belonging to the input vector set of the mode with the higher power consumption. In the previous example, these vectors are the ones that generate the right most lobes in the distribution graph. An example test-case proving the aforementioned theory is an industrial IC. The design has been implemented in a 45nm Process Design Kit (PDK). The library file (.lib) was precharacterized in 8 different voltage corners from the nominal voltage 0.9V to 0.76V with step 0.02V at 25C using Synopsys SiliconSmart. The design operates in 4 different modes with period 1.6ns and 8.0ns. The size in terms of instances for the design was 322632 std-cells. The tool simulated the IC for 2500 periods in the 4 modes of operation, using 4 different SAIF files for stimulus, SPEF file for the extracted interconnects and SPICE file for the extracted Power Supply Network. The PDN had 10397 vias connecting M1 with the rest of the PDN. For convergence on the voltage/current waveforms, 3 iterations were enough.

Table-1 demonstrates, for each mode of operation of the IC (Column-1), the corresponding results for power consumption (Column-2) along with the worst case voltage-drop (Column-3) and ground-bounce (Column-4). The leakage was calculated in 357.04 mW.

Table 2 NANOPOWER™- Power Integrity simulation results for each Mode of operation.

Mode	Average Power (mW)	Worst Case Voltage (VDD nominal : 0.9V)	Worst Ground Bounce (VSS nominal : 0.0V)
Mode-1	513.12	0.8895 V	0.0264 V
Mode-2	609.11	0.8682 V	0.0480 V
Mode-3	680.98	0.8490 V	0.0552 V
Mode-4	741.23	0.8405 V	0.0697 V

Runtime: The tool used 12 threads for parallelization. Each simulation period, depending on the switching activity, needed ~10-11mins to finish for the modes of operation with 6.4ns period and ~4 - 4.5mins for the modes of operation with 1.6ns period. The results of the statistical prediction engine (EVT) are depicted in the following Table.

Runtime: EVT module ran using 20 threads for parallelization and took ~2 hours to finish processing for the modes with 1.6ns period and ~7 hours for the modes with 6.4ns period.

Table 3 NANOPOWER™ - Power Integrity Statistical Prediction Engine results.

Mode	EVT-estimation (VDD nominal 0.9V)	%Voltage-drop (VDD nominal 0.9V)	EVT-estimation (VSS nominal 0.0V)	%Ground Bounce (VSS nominal 0.0V)
Mode-1	0.8601 V	4.444%	0.0307 V	3.411%
Mode-2	0.8568 V	4.800%	0.0511 V	5.677%
Mode-3	0.8019 V	10.900%	0.0690 V	7.667%
Mode-4	0.7844 V	12.845%	0.0748 V	8.311%

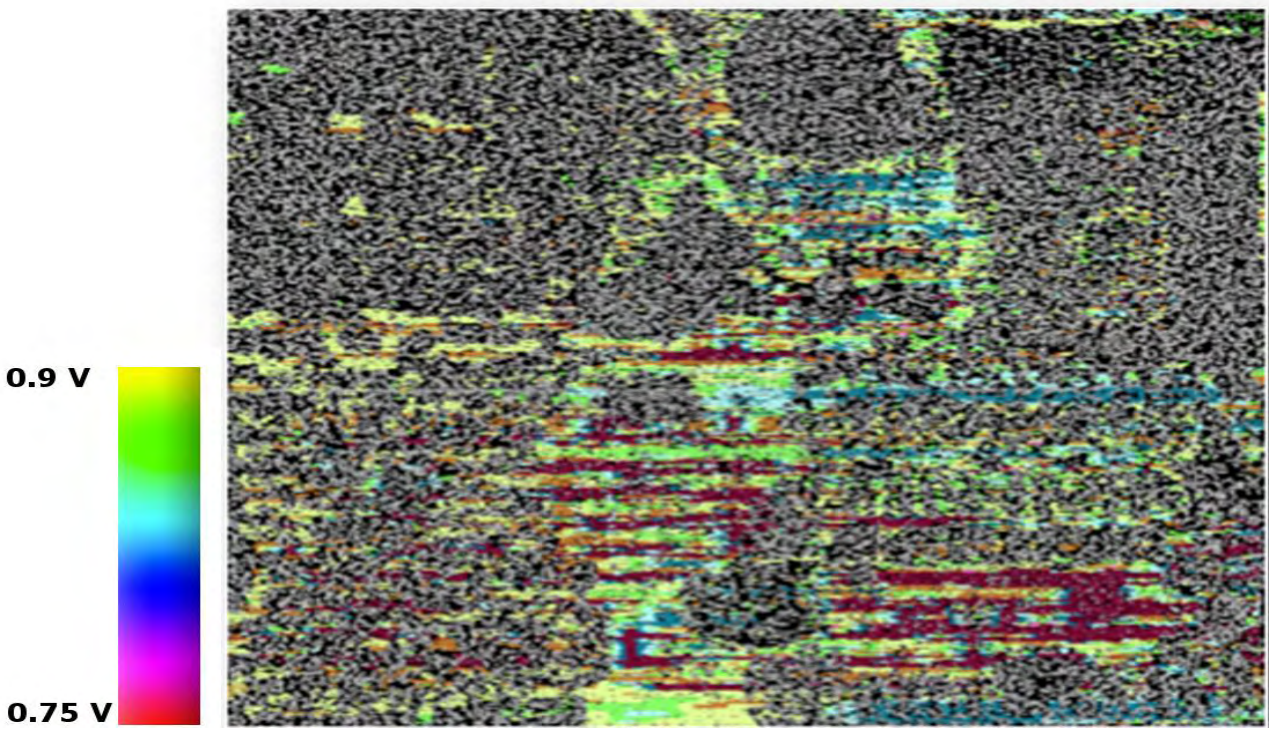


Figure 23 Mode-1 Voltage-drop colormap using EVT results (Magma-Talus).

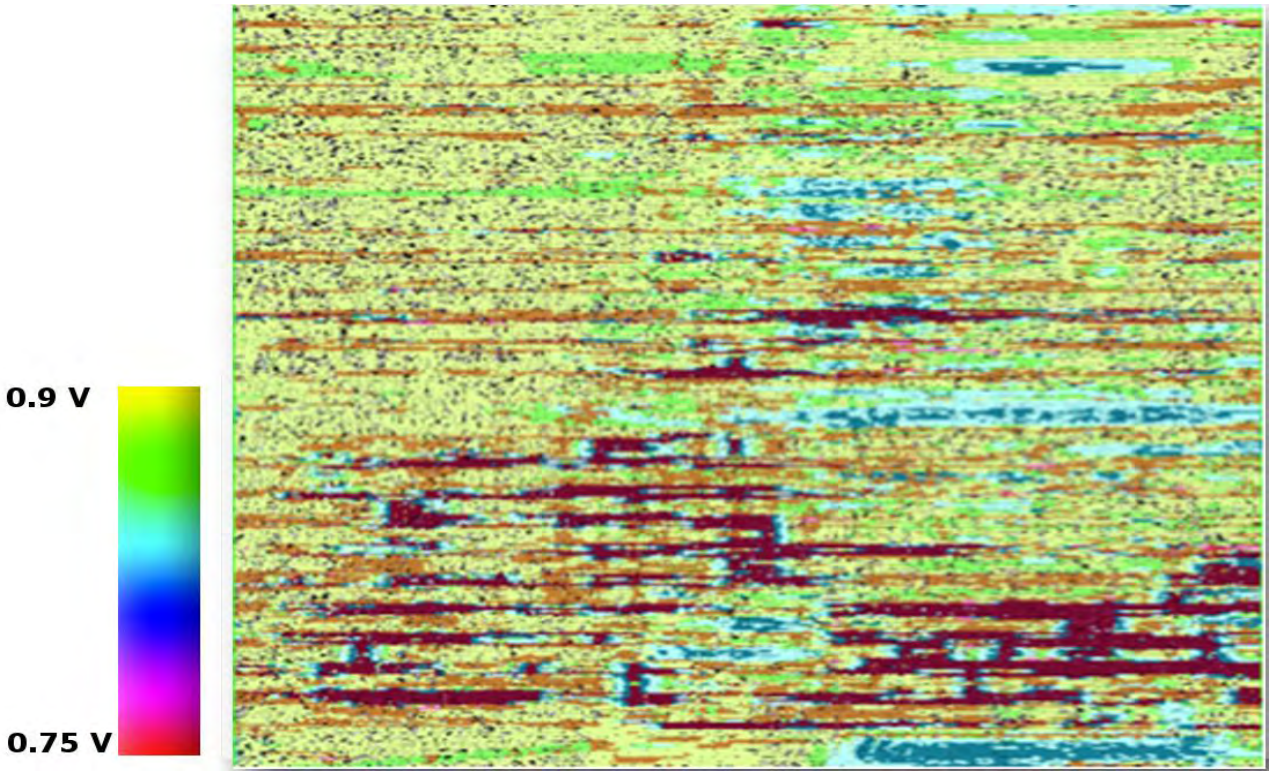


Figure 24 Mode-2 Voltage-drop colormap using EVT results (Magma-Talus).

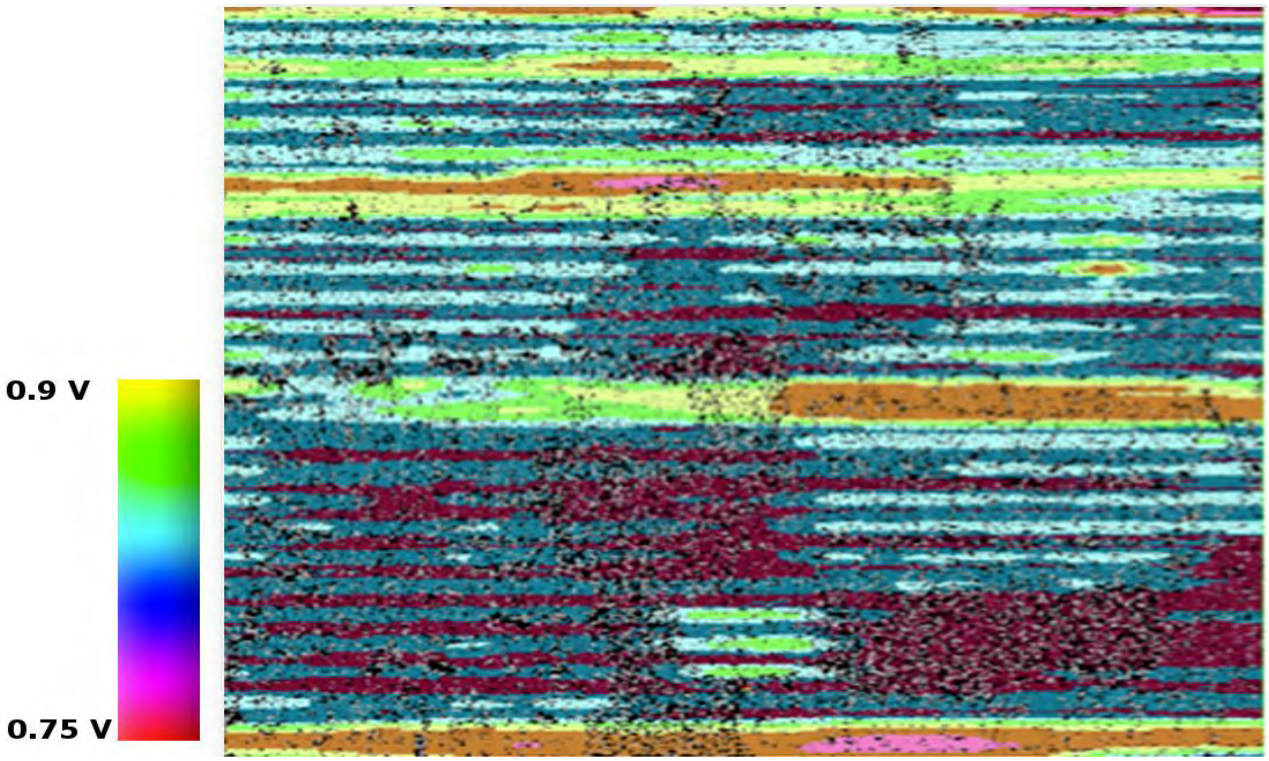


Figure 25 Mode-3 Voltage-drop colormap using EVT results (Magma-Talus).

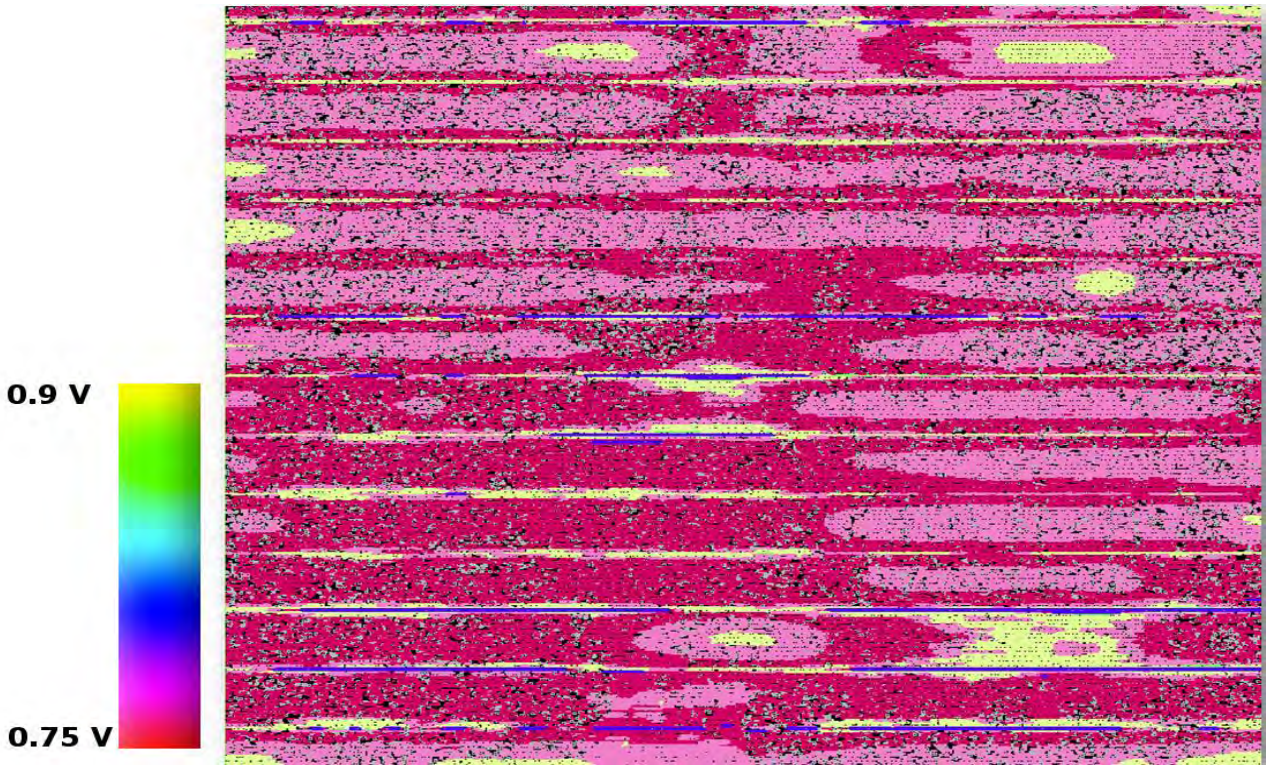


Figure 26 Mode-4 Voltage-drop colormap using EVT results (Magma-Talus).

The resulted colormaps (**Figure 23**, **Figure 24, Figure 25, and Figure 26**) prove that the power mode with the highest power consumption, in other words the right most lobe in the specific current distribution graph and thus the mode of operation with the highest switching activity is far more likely to report the worst-case voltage drops in the IC. One may observe in aforementioned colormaps a common pattern at the areas of the design which implies that independently of the mode the design operates there are areas that will produce higher voltage drop compared to the rest of the IC. This may be an outcome of a badly designed power grid in the specific areas, high switching activity of the cells under this area or a combination of the switching activity and the power grid. Of course this conclusion is not an inviolable rule since there may be cases where a combination of highly power consuming block and a bad power grid may result locally in a design to a very high voltage drop using an input vector pair with low switching activity and thus lower total power consumption.

6 Timing Analysis

6.1 General description

Timing analysis is the methodical analysis of a digital circuit to determine if the timing constraints imposed by the initial design specifications are met. Meeting the initial specifications means that the design meets all setup, hold, and pulse-width times' requirement. Anything else can be compromised but not timing! The basic features of the Timing Analysis are the "Clock" and the "Sequential" elements of the design (Flip-flops, Latches). The "Clock" determines the operating speed of the IC. There are two types of Timing analysis, Static Timing Analysis (STA) which is the most common and widespread in the field and Dynamic Timing Analysis (DTA). These are some of the basic requirements; an IC should meet at the end of Timing Closure process of a design.

Related to Clock:

- Glitch free
- All generated Clocks should be:
    - Clean.
    - Bounded in a specific period and duty cycle.
    - Have a known relationship to other clocks of interest in the design.
- Meet the minimum pulse width for both high and low phases.
- Maximum jitter in case of PLLs.
- Calculate the data "passing" from one clock edge to the other using the worst case duty cycle.

Related to Sequential elements:

- Make sure that all parameters of flip-flops always met. The only exception is when synchronizers are used to synchronize asynchronous signals
- For asynchronous presets and clears, there are two basic parameters (Recovery and Removal) must be met.
- All setup and hold times are met for the earliest/latest arrival times for the clock.
- Setup times are generally calculated by designers and suitable margins can be demonstrated under test. Hold times, however, are frequently not calculated by designers.
- When passing data from one clock domain to another, ensure that there is either known phase relationships which will guarantee meeting setup and hold times or that the circuits are properly synchronized.

6.2 Static Timing Analysis (STA)

Static Timing Analysis is a method which validates the performance in terms of timing for a digital design. The method performs calculations, without using input or output vectors, resulting to the delay of each path of a design, using the worst possible delay for each digital element, ensuring this way that no timing violations exist for all the paths, under the worst-case conditions. STA, checks the design for valid timing but it does not perform any validation for the functional operation of the design. Static timing analysis seeks to answer the question: "Will the correct data be present at the data input of each synchronous device and the design primary outputs when the clock edge arrives, under all possible conditions?" In the term "Static Timing Analysis", the word "Static" alludes to the fact that the analysis is carried out in an input-independent way. In modern ICs, which are enormous and complex, there are huge numbers of logic paths and STA performs timing analysis on all possible logic paths. However, it is worth noting that STA is not suitable for all design styles. It has proven efficient only for fully synchronous designs. Since the majority of chip design is synchronous, it has become a mainstay of chip design over the last few decades. In designs there are four general categories of Timing Paths, categorized by the type of signals, each having a "Start-Point" (Input port or sequential element) and an "End-Point" (Output port or sequential element) and a calculated value for the delay of the path. Start-Point and End-Point are different depending on the type of path. Following is a brief description of the four Timing Path categories.

6.2.1 Design Timing Path Types

The following sub-sections describe all possible timing paths [7], which are subject of Timing Analysis and may exist in all modern ICs. The timing paths are the main outcome of this Analysis along with the delay.

6.2.1.1 Data Path

This path-category has 4 different types resulting from the combination of 2 types of Start-Point and 2 types of End-Point. The possible Start-Points and End-Points are:

- Start-Point
  - Input port of the design (because the input data can be launched from some external source).
  - Clock pin of the flip-flop/latch/memory (sequential element)
- End-Point
  - Data input pin of the flip-flop/latch/memory (sequential element)
  - Output port of the design (because the output data can be captured by some external sink).

The resulting Data Path types are the following:

1. PATH-A: starts from an input port and ends to the data input of a sequential element. (Input to Register).
2. PATH-B: starts from the clock pin of a sequential element and ends to the data input of a sequential element. (Register to Register).
3. PATH-C: starts from the clock pin of a sequential element and ends to an output port. (Register to Output).
4. PATH-D starts from an input port and ends at an output port. (Input to Output).

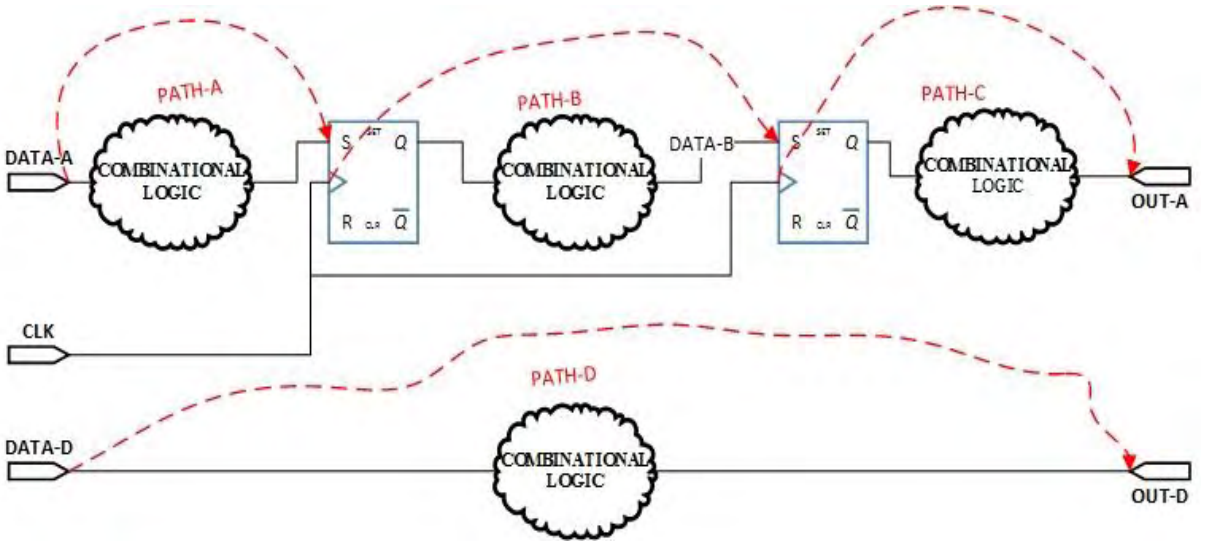


Figure 27 Timing Analysis - Data Path Types.

6.2.1.2 Clock Path

This category's paths start from the input port/pin of the design which represents the Clock input and the end point is the Clock pin of a sequential element. In between the Start Point and the End Point there may be lots of Buffers/Inverters/clock dividers. The possible Start-Points and End-Points are:

- Start-Point
  - Clock input port.
- End-Point
  - Clock pin of the flip-flop/latch/memory (sequential element).

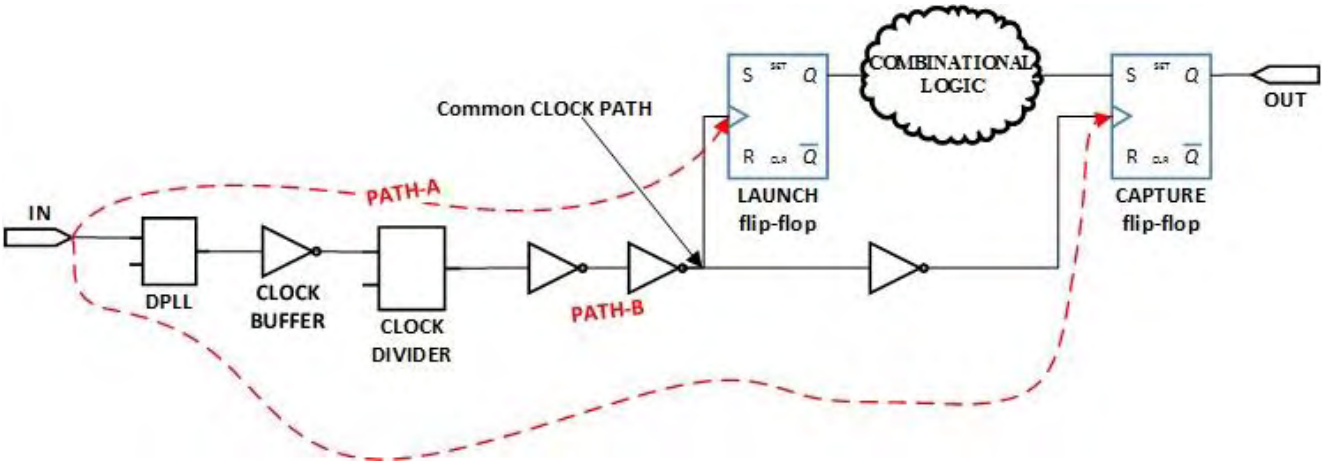


Figure 28 Timing Analysis - Clock Path.

6.2.1.3 Clock Gating Path

This category's paths may pass through "gated elements" to achieve additional advantages. In this case, characteristics and definitions of the clock change accordingly. We call this type of clock path as "gated clock paths". Start-Point (LD pin in figure) is not part of any clock but it is used for gating the original Clock signal (CLK in figure). Such type of paths is neither part of Clock path nor Data Path, due to the Start Point and End Point definition of these paths. Thus such types of paths are part of "Clock Gating Path". The possible Start-Points and End-Points are:

- Start-Point
  - Input port of the design.
- End-Point
  - Input port of clock-gating element.

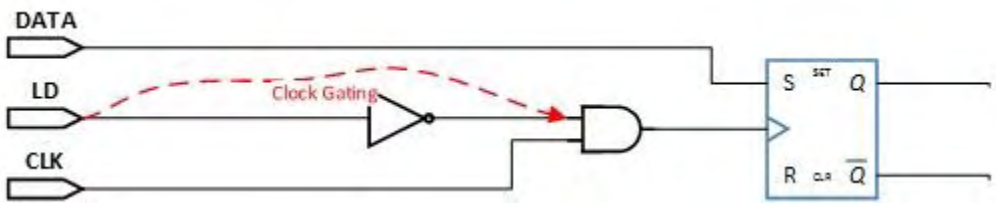


Figure 29 Timing Analysis - Clock Gating Path.

6.2.1.4 Asynchronous Path

This category includes paths from an input port to an asynchronous "set", "reset" or "clear" pin of a sequential element. The functionality of "set" and "reset" pins is independent from the clock edge and may be triggered without any dependence to the Clock, the path may start functioning at any time. Since the path is not synchronized with the rest of the design it is called Asynchronous Path. The possible Start-Points and End-Points are:

- Start-Point
  - Input Port of the design.
- End-Point
  - Set/Reset/Clear pin of the flip-flop/latch/memory (sequential element).

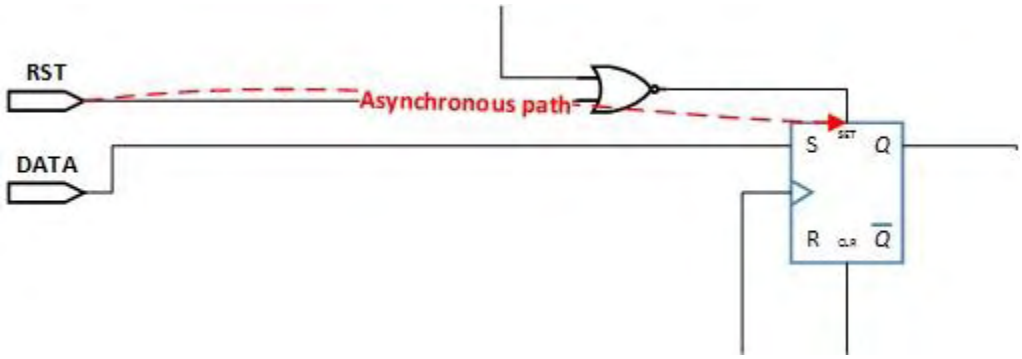


Figure 30 Timing Analysis - Asynchronous Path**.**

6.2.2 Reported Timing Path Types

During Timing Analysis there are few Timing Paths which are sub-set of the aforementioned Design Timing Paths but with specific characteristics as described below.

6.2.2.1 Critical Path

It is the path in the design which creates the largest delay.

- Critical paths are timing-sensitive functional paths. The timing of these paths is critical, thus no additional gates are allowed to be added to these paths, in order to prevent increasing their delay.
- Timing critical path are those paths that may violate design's timing constraints. What normally happens is that after synthesis the tool will give you a number of paths which have a negative slag. The first thing you would do is to make sure those paths are not false or multicycle since in these cases you can just ignore them.

Taking a typical example (in a very simpler way), the STA tool will add the delay contributed from all the logic connecting the Q output of one flop to the D input of the next (including the CLK->Q of the first flop), and then compare it against the defined clock period of the CLK pins (assuming both flops are on the same clock, and taking into account the setup time of the second flop and the clock skew). This should be strictly less than the clock period defined for that clock. In case the reported delay is less than the clock period, then the "path meets timing". If it is greater, then the "path fails timing". The "critical path" is the path out of all the possible paths that either exceeds its constraint by the largest amount, or, if all paths pass, then the one that comes closest to failing.

6.2.2.2 False Path

- Physically exist in the design but those are logically/functionally incorrect path. Means no data is transferred from Start Point to End Point. There may be several reasons of such path present in the design.
- Some time we have to explicitly define/create few false paths with in the design. e.g. for setting a relationship between 2 Asynchronous Clocks.
- The goal in static timing analysis is to do timing analysis on all "true" timing paths, so these types of timing paths are excluded from timing analysis.
- Since false paths are not exercised during normal circuit operation, they typically do not meet timing specification, considering false path during timing closure can result into timing violations and the procedure to fix would introduce unnecessary complexities in the design.
- There may be few paths in the design which are not critical for timing or masking other paths which are important for timing optimization, or never occur with in normal situation. In such case, to increase the run time and improving the timing result, sometimes the designer has to declare such path as a False path , so that Timing Analysis Tool ignore these paths and so the proper analysis with respect to other paths or during optimization don't concentrate over such paths. One example of this e.g. "A" path between two multiplexed blocks that are never enabled at the same time. You can see the following figure for this.

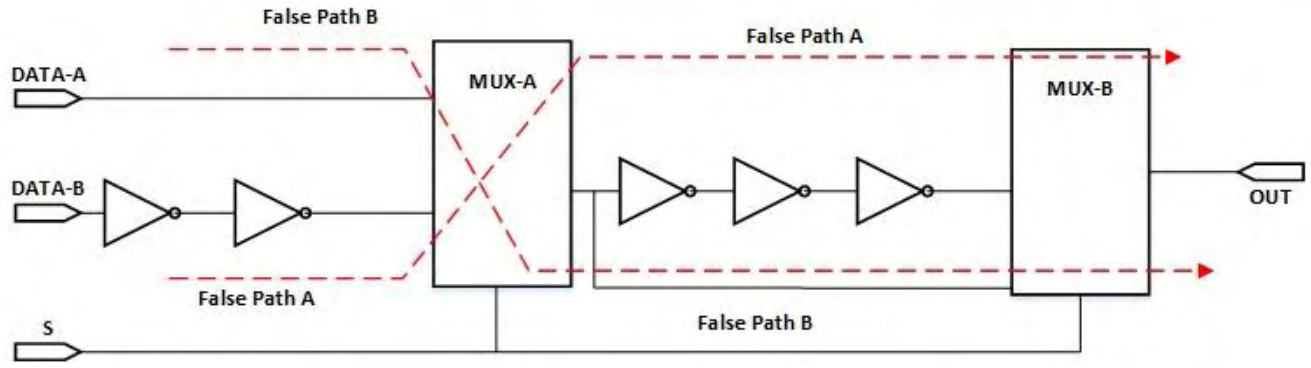


Figure 31 Timing Analysis - False Path.

Here the False path 1 and False Path 2 cannot occur at the same time but during optimization it can affect the timing of another path. So in such scenario, we have to define one of the paths as false path. Not all paths that exist in a circuit are "real" timing paths. For example, let us assume that one of the primary inputs to the chip is a configuration input; on the board it must be tied either to VDD or to GND. Since this pin can never change, there are never any timing events on that signal. As a result, all STA paths that start at this particular Start Point are false. The STA tool (and the synthesis tool) is not aware that this pin is going to be tied off, so it needs to be told that these STA paths are false, which the designer can do by setting a "false-path" directive. When setting the paths to be false, the STA tool will not analyze it (and hence will not compare it to a constraint, so this path cannot fail), nor will a synthesis tool do any optimizations on that particular path to make it faster; synthesis tools try and improve paths until they "meet timing" - since the path is false, the synthesis tool has no work to do on this path.

Thus, a path should be declared false if the designer is aware that the path in question is not a real timing path, even though it looks like one to the STA tool. Designers should be very careful with declaring a path false. If you declare a path false, and there is a case where it is actually a real path, then you have created the potential for a circuit to fail, and for the most part, you will not catch the error until the chip is on a board, and (not) working. Typically, false paths exist:

- From configuration inputs like the one described above.
- From "test" inputs; inputs that are only used in the testing of the chip and are tied off in normal mode (however, there may still be some static timing constraints for the test mode of the chip).
- From asynchronous inputs to the chip (and you must have some form of synchronizing circuit on this input) (this is not an exhaustive list, but covers the majority of legitimate false paths).

So we can say that false paths should not be derived from running the STA tool (or synthesis tool); they should be known by the designer as part of the definition of the circuit, and constrained accordingly at the time of initial synthesis.

6.2.2.3 Multicycle Path

A multicycle path is a timing path that is designed to take more than one clock cycle for the data to propagate from the Start Point to the End Point.

A multicycle path is a path that is allowed multiple clock cycles for propagation. Again, it is a path that starts at a timing Start Point and ends at a timing End Point. However, for a multi-cycle path, the normal constraint on this path is overridden to allow for the propagation to take multiple clocks. In the simplest example, the Start Point and End Point are flops clocked by the same clock. The normal constraint is therefore applied by the

definition of the clock; the sum of all delays from the CLK arrival at the first flop to the arrival at the D input of the second clock should take no more than 1 clock period minus the setup time of the second flop and adjusted for clock skew. By defining the path as a multicycle path you can tell the synthesis or STA tool that the path has N clock cycles to propagate; so the timing check becomes "the propagation must be less than N x clock period, minus the setup time and clock skew". N can be any number greater than 1.

Few examples are:

- When designers are doing clock crossing from two closely related clocks; i.e. from a 30MHz clock to a 60MHz clock
    - Assuming the two clocks are from the same clock source (i.e. one is the divided clock of the other), and the two clocks are in phase.
    - The normal constraint in this case is from the rising edge of the 30MHz clock to the nearest edge of the 60MHz clock, which is 16ns later. However, if you have a signal in the 60MHz domain that indicates the phase of the 30MHz clock, you can design a circuit that allows for the full 33ns for the clock crossing, then the path from flop30 -> to flop60 is a MCP (again with N=2).
    - The generation of the signal 30MHZ_is_low is not trivial, since it must come from a flop which is clocked by the 60MHz clock, but show the phase of the 30MHz clock.
- Another place would be when there exist different parts of the design that run at different, but related frequencies. Again, consider a circuit that has some stuff running at 60MHz and some running on a divided clock at 30MHz.
    - Instead of actually defining 2 clocks, you can use only the faster clock, and have a clock enable that prevents the clocks in the slower domain from updating every other clock.
    - Then all the paths from the "30MHz" flops to the "30MHz" flops can be MCP.
    - This is often done since it is usually a good idea to keep the number of different clock domains to a minimum.

6.2.2.4 Single Cycle Path

A Single-cycle path is a timing path that is designed to take only one clock cycle for the data to propagate from the Start Point to the End Point.

6.2.2.5 Launch Path and Capture Path

Both timing path types are inter-related. When a flip flop to flip-flop path such as *Flip-Flop A* to *Flip-Flop C* is considered, one of the flip-flop launches the data and other captures the data. So here *Flip-Flop A* is referred to "launch Flip-flop" and *Flip-Flop C* referred to "capture flip-flop".

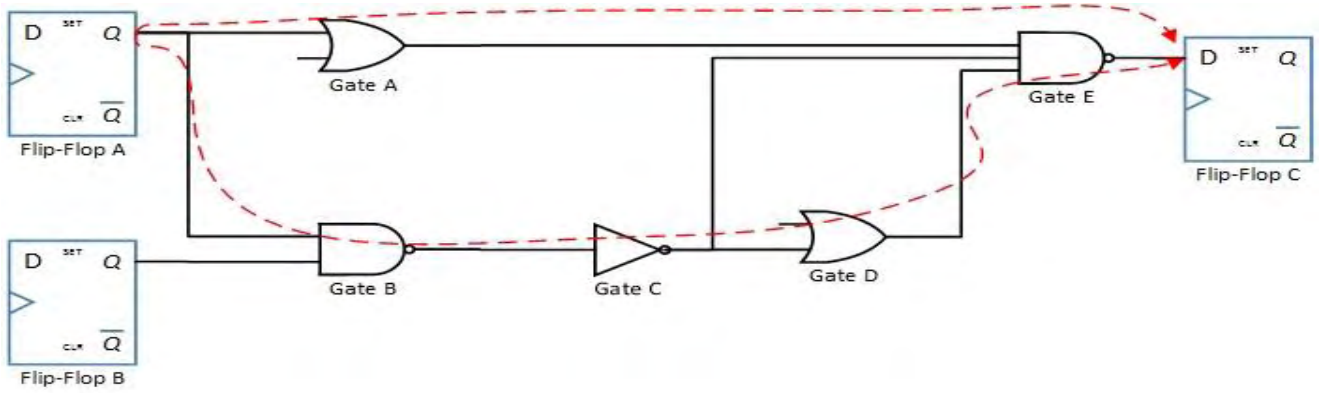


Figure 32 Timing Analysis - Launch flip-flop and Capture flip-flop.

These Launch and Capture terminology are always referred to a flip-flop to flip-flop path. Means for this particular path (*Flip-Flop A* to *Flip-Flop C*), *Flip-Flop A* is launch flip-flop and *Flip-Flop C* is capture flip-flop. Now if there is any other path starting from *Flip-Flop C* and ends to some other flip-flop (let's assume *Flip-Flop D*), then for that path *Flip-Flop C* become launch flip-flop and *Flip-Flop D* be as capture flip-flop. The name "Launch path" referred to a part of clock path. Launch path is launch clock path which is responsible for launching the data at launch flip flop. Similarly Capture path is also a part of clock path. Capture path is capture clock path which is responsible for capturing the data at capture flip-flop.

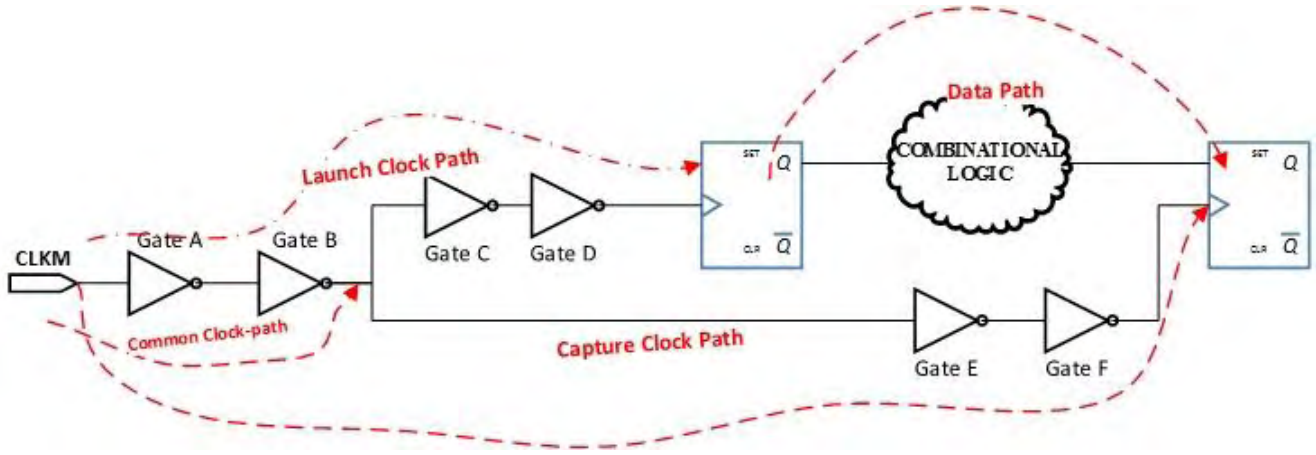


Figure 33 Timing Analysis – Launch Clock Path and Capture Clock Path.

In this example *Flip-Flop A* is referred to launch flip-flop and *Flip-Flop B* as capture flip-flop for "Data path" between *Flip-Flop A* to *Flip-Flop B*.So Start point for this data path is *Flip-Flop A* /CK and End Point is *Flip-Flop B* /D.

- Launch path and data path together constitute arrival time of data at the input of capture flip-flop.
- Capture clock period and its path delay together constitute required time of data at the input of capture register.

Capture and Launch paths correspond to Data paths. This means that the same clock path can be a launch path for one data path and be a capture path for another Data-Path.

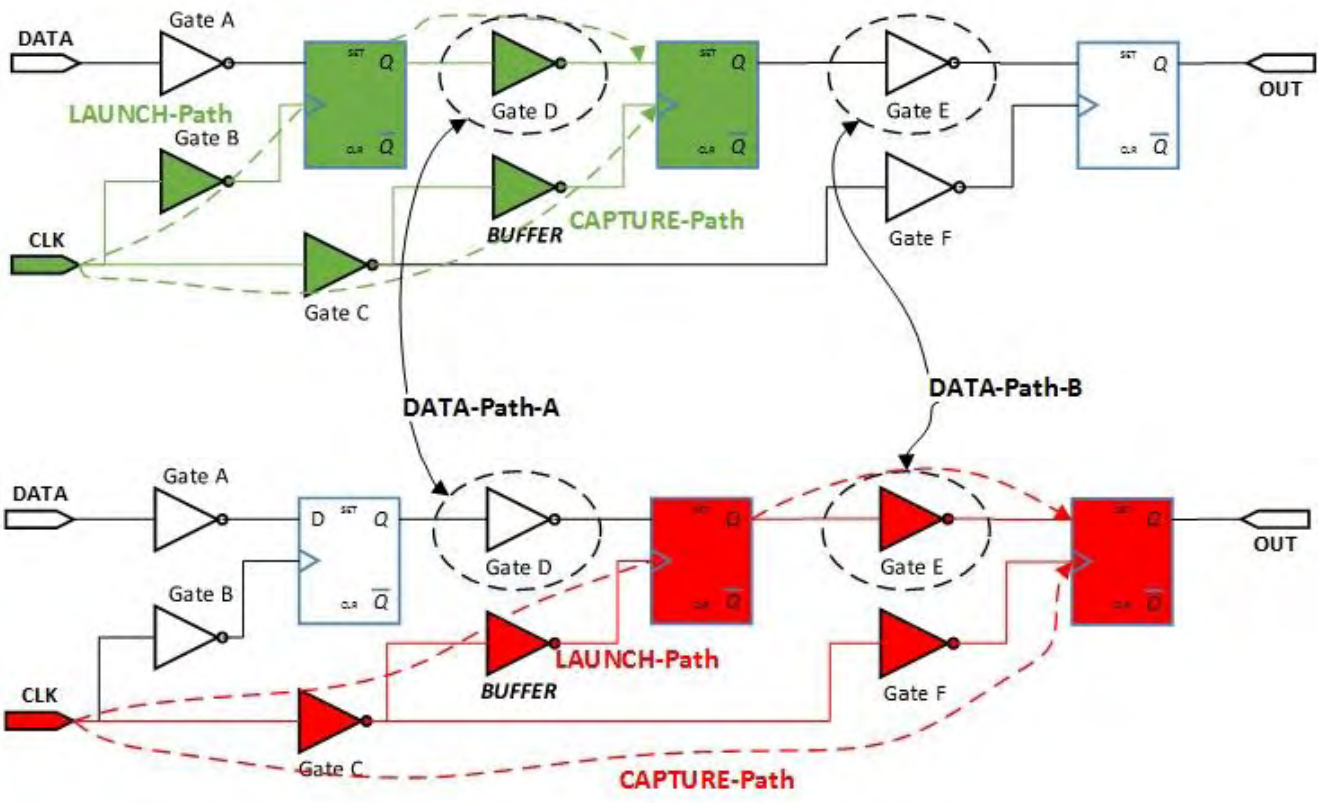


Figure 34 Timing Analysis – Launch Path and Capture Path.

Same clock path behave like **Capture-Path** and **Launch-Path** for different DATA Path. Here you can see that for **DATA-Path-A** the clock path through ***BUFFER*** cell is a capture path but for **DATA-Path-B** it's a **Launch-Path**.

6.2.2.6 Longest and Shortest Path

Between any 2 nodes, there can be many paths. Longest path is the one that takes longest time; this is also called worst path or late path or a max path. The shortest path is the one that takes the shortest time; this is also called the best path or early path or a min path. In the figure below, the longest path between the 2 flip-flops is through the cells ***BUFFER***, ***GATE-A*** and ***GATE-B***. The shortest path between the 2 flip-flops is through the cell ***GATE-B***.

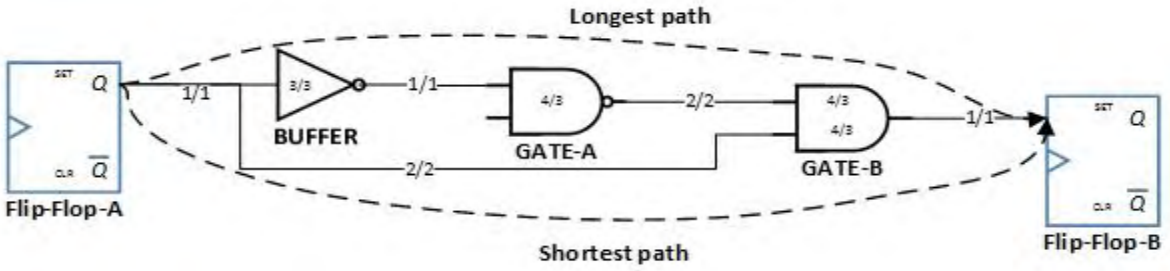


Figure 35 Timing Analysis – Longest Path and Shortest Path.

6.3 Dynamic Timing Analysis (DTA)

Dynamic Timing Analysis performs a functional verification of the design by applying input vectors in the design and checking for the correct output vectors which arrive at the design outputs within the required time specified by the designer [61]. This approach is an extension of simulation and ensures that circuit timing is tested in its functional context. This method reports timing errors that functionally exist in the circuit and avoids reporting errors that occur in unused circuit paths. The most common dynamic timing analysis is the so-called min-max analysis method. Under min-max timing analysis, both minimum and maximum delays of circuit components are used to generate outputs, which are ranges (the spread of earliest data and latest arrival data) instead of edges. Since outputs are in turn fed into inputs, managing the ranges (merging them) can become very complex. As can be seen, if both min version & max version of the delays must be used, the simulation speed will be extremely slow. Another major issue with dynamic timing analysis is the incomplete coverage. It may only check circuitry that is exercised by test stimulus, which may leave critical paths untested, and timing problems undiscovered. It is also not path oriented. Since dynamic timing analysis reports errors on a certain pin at a certain time, the user must trace through the schematic to locate the path that caused the problem (difficult for large designs). Finally this method requires development time for test vectors. Dynamic timing analysis tools often track more information than logic simulators, making their performance slower. Also each component must contain both timing information and a functional model before timing verification can proceed. This could prevent the use of new parts that do not have functional models. It should be noted that min-max simulation is not currently used in the industry. Instead, either functional simulation with timing (timing simulation) or formal verification method is typically used to verify complex IC designs. Typically people use the max version of delays to verify the circuit works under worst-case timing (no setup issues) and min version of the delays to verify best-case timing (no hold issues). Dynamic timing analysis extends logic simulation by reporting violations in terms of simulation times and states. To test circuit timing using worst-case conditions, dynamic timing analysis evaluates the circuit using minimum and maximum propagation delays for each component for each component in the design. Since dynamic timing analysis performs a simulation, it can use the same stimulus as a logic simulation. Because the stimulus functionally exercises the design, false errors of unused or uninteresting paths are not tested. Note a timing simulation reports results differently than a logic simulation. A logic simulation reports results as edge times and a timing simulation reports results as regions of ambiguity. The results of a timing simulation do not specify exactly when an event occurs; they specify a range of time in which an event can occur.

6.3.1 ATPG

6.3.1.1 General Description

ATPG stands for Automatic Test Pattern Generation. ATPG is a process used in Electronic Design Automation field in order to automatically generate input vectors or input patterns that will be used to check a digital design for faults. The generated input patterns aim to ascertain presence or absence of fault(s) at some location(s) in a circuit. The vectors are sequentially applied to the design under test and the designs outputs for each set of inputs is compared with the expected outputs. Differences between the simulation output pattern and the expected output pattern of the design means that it is faulty. The effectiveness of the ATPG is measured primarily by the fault coverage achieved and the cost of performing the test. The process is divided in two basic steps, the first one is the creation of the test and the second one is the application of the test. The goal of an ATPG is to be efficient in memory, space and time requirements. Thus, the process should generate the smallest possible set of vectors needed to detect all the important faults of the design. The main considerations are:

1. Construction time for the smallest possible input pattern.
2. Size in terms of hardware/software system needed to run the ATPG process.
3. The size in terms of memory for the testing process itself.
4. The time needed to load and run the generated input patterns.
5. Any external equipment that might be required.

The creation of input pattern is a mathematical process that may result by the following general methods:

1. Manual methods
2. Algorithmic methods (with or without heuristics)
3. Pseudo-Random methods.

6.3.1.2 Existing Algorithms

In VLSI designs, testing with a high “fault” coverage is an extremely challenging process due to the complexity of the design. Therefore, several different ATPG methods have been developed to address combinational and sequential circuits. The main methods are separated into two basic categories, the deterministic where the input pattern is systematically developed with a definite outcome for the targeted faults being computationally intensive and the probabilistic where the input pattern is generated by 'chance' and simply confirmed by fault simulations for effectiveness:

- **Deterministic Algorithms**
    - The **D Algorithm** was the first practical test generation algorithm in terms of memory requirements and was proposed by Roth back in 1966. It introduced **D and D'** notation**,** which continues to be used in most ATPG algorithms. **D** simply stands for a **'1'** in a good design and a **'0'** in a faulty one. On the other hand, **D'**, which is the opposite of **D**, stands for a **'0'** in a good circuit and **'1'** in a faulty design. Thus, propagating a **D** or **D'** from the inputs to the output simply means applying a set of input vectors to a device to make its output exhibit an 'error' if a fault within the circuit exists.
    - The **PODEM Algorithm (**Path-Oriented Decision Making) is an improvement over the D Algorithm and was created back in 1981, by Prabhu Goel, to address an issue that D algorithm had with XOR gates. PODEM was the first major enhancement to the D algorithm as far as efficiency is concerned. The complexity of D algorithm increases exponentially with the

number of internal circuit nodes, while PODEM's complexity increases exponentially with the number of circuit inputs. PODEM is more efficient than the D algorithm due to the fact that the number of inputs is usually much smaller than the number of internal circuit nodes.

- The **FAN Algorithm** (Fan-Out Oriented) is an enhanced version over PODEM, which limits the ATPG search space to reduce computation time and accelerates backtracking. It also utilizes circuit topology information to increase search efficiency
- The **SAT** (Satisfiability/Propositional Satisfiability) algorithms, based on Boolean satisfiability are used to generate test vectors. SAT is the first problem that was proven to be *NP-complete*. The algorithms checks if there exists an interpretation that satisfies a given Boolean expression. This means that the algorithm searches for combinations of values (True/False) for the variables of the expression that in the case they are assigned the expression evaluates to either True (satisfiable expression) or False (unsatifiable expression).

- **Probabilistic Algorithms**
  - **Pseudorandom Test Generation** method, which is the simplest one to create input vectors and uses a pseudorandom number generator for the input pattern generation and relies on logic simulation to compute good machine results, and fault simulation to calculate the fault coverage of the generated vectors.
  - **WASP** (Wavelet Automatic Spectral Pattern Generator) which is an improvement over spectral algorithms for sequential ATPG. It uses wavelet heuristics to search space to reduce computation time and accelerate the compactor.

7 Voltage-Drop Aware Timing Analysis

Continuously shrinking transistor sizes, decreasing power supply nominal voltages, increasing design sizes (leading to higher power consumption) and the rising operating frequencies aggravate the Voltage-drop on the power delivery network of ICs [8] [9]. Power supply noise, as we have already seen in all previous sections of this Thesis, results from the relation between the impedance of the Power Delivery Network and the amount of current drawn by the standard-cells and macros each moment of operation of an IC. Since power supply noise effect has significant impact on the performance of deep submicron designs [10] [11], the challenge of accurately estimating the worst case delay that may occur during IC operation, limiting the performance of an IC, becomes an issue of high priority. In modern deep submicron ICs the power grid cannot be designed independently from the rest of the design and the main reason is, that the power grid and timing are certainly no longer considered decoupled. This is not an issue that can safely be engineered by just having timing margins. Designers have to analyze the behavior of the power grid at the critical timing corners rather than analyze the power grid and power just from the perspective of what is the peak power for this design and which the largest di/dt events are. Moreover, designers have to analyze the behavior of the power grid with respect to timing, and timing with respect to the power grid. Back in 2000 the coupling of functionality and timing was an intractable problem. For several years solutions came from P&R algorithms but nowadays, when the designs are much larger and much more complex, these tightly coupled phenomena have become a major bottleneck for the design closure and the existing P&R algorithms are not able to tackle this issue successfully in the early stages of the design cycle. The solution has to come from the back-end of the design flow, where there is enough information for the designers about the IC's functionality and the parasitics.

7.1 Voltage-Drop Impact on Timing

As voltage level on the power pin of the design decreases, the delay of each gate or block in the design needs more time to load or offload both the internal capacitances and the output capacitances from the interconnects and the pins of the gates or blocks it drives. In a broader view, the partial delay of each gate cumulatively may result in a large delay on the timing closure. Another extremely important and at the same time hidden impact of the Voltage-drop on timing is the structure of the Critical Path itself. To demonstrate the impact of Voltage-drop on timing, two simple experiments but sufficient enough to achieve this goal have been set. The first experiment demonstrates the impact of Voltage-drop on the total delay of the design and the relation between the voltage level and the delay. The second experiment reveals the impact of the Voltage-drop on the structure of the Critical Paths reported by industrial STA tools.

7.1.1 Voltage-drop impact on Delay

In the first experiment, we demonstrate the impact of voltage drop in the total delay, for a small ISCAS design (C1355 in NanGate45nm). To do so, we have run STA (using Synopsys NanoTime®) consecutive times; sweeping the reference voltage in each run, from the PDK's default voltage (1.0V) to reach 50% voltage-drop, with a small step. The results are demonstrated through the following graphs (**Figure 36** and **Figure 37**). In the first plot, Y axis is the reference power supply in Volts while in X axis is the total delay (STA output) in ps. As one may infer from the graph the delay increases in a quadratic way as the voltage drops. The second graph gives the relation between the Voltage-drop percent (%) and the multiples of the delay corresponding to the PDK's nominal voltage. As one may observe a delay corresponding to 50% of the nominal voltage is ~4 times the delay corresponding to the nominal voltage for the design under test.

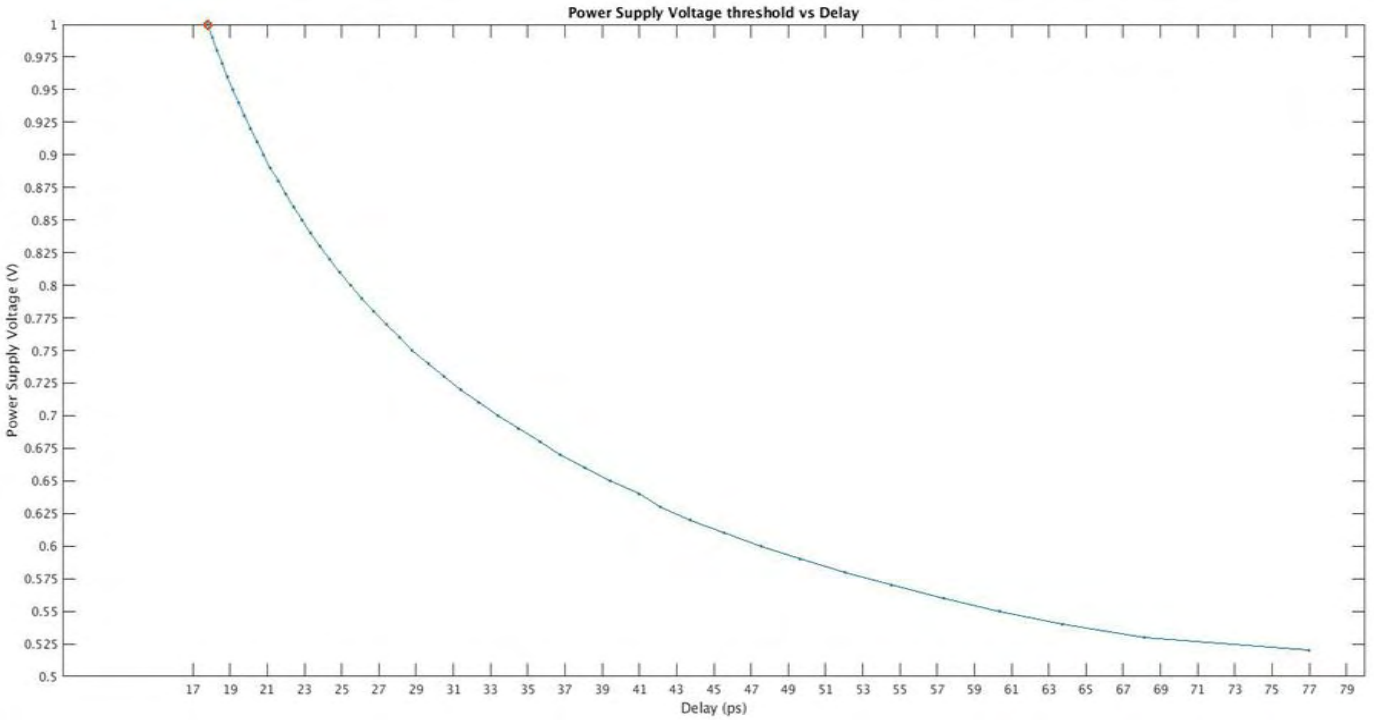


Figure 36 Power Supply Level vs Delay (STA).

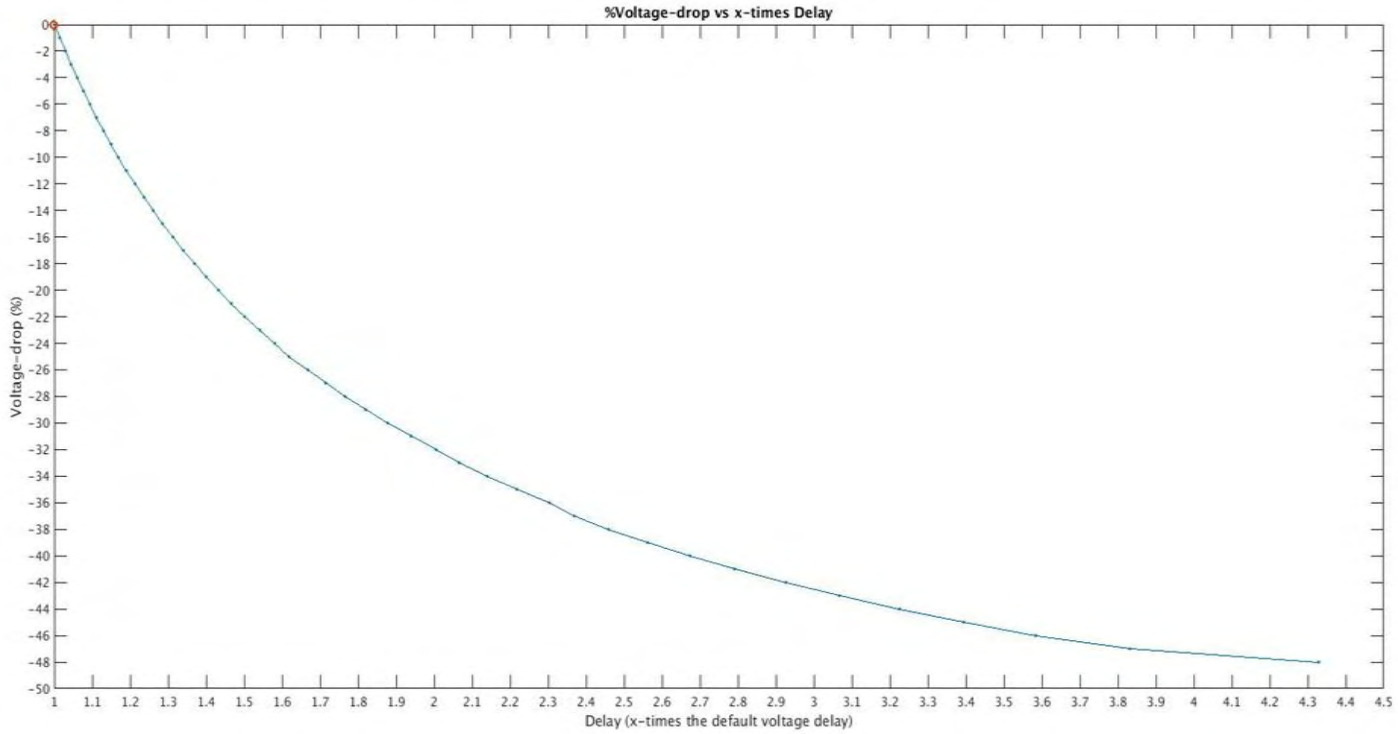


Figure 37 %Voltage-drop vs times nominal voltage Delay (STA).

7.1.2 Voltage-drop impact on Critical Path structure

The second experiment demonstrates the way reduction of voltage level on the power supply network leads STA to report not just a different delay but also a different critical path. A small ISCAS design (C1355 in NanGate45nm) is used to run STA (using Synopsys NanoTime®) twice, configuring the tool to report also the Critical Path. The differences in the structure of the reported Critical Path are visible in the Table in ***Appendix D*** where the first column (STA 1.1V) includes the gates comprising the reported Critical Path using the nominal voltage, 1.1 Volt in STA and the second column including the reported Critical Path using 0.9 Volt in STA. Another experiment is the annotation on JPEG design of different voltage levels for STA. The locations of the corresponding Critical Paths, from the different STA runs are depicted in **Figure 38**. It is profound that Voltage-drop not only changes the worst-case delay but also the structure of the reported Critical Path.

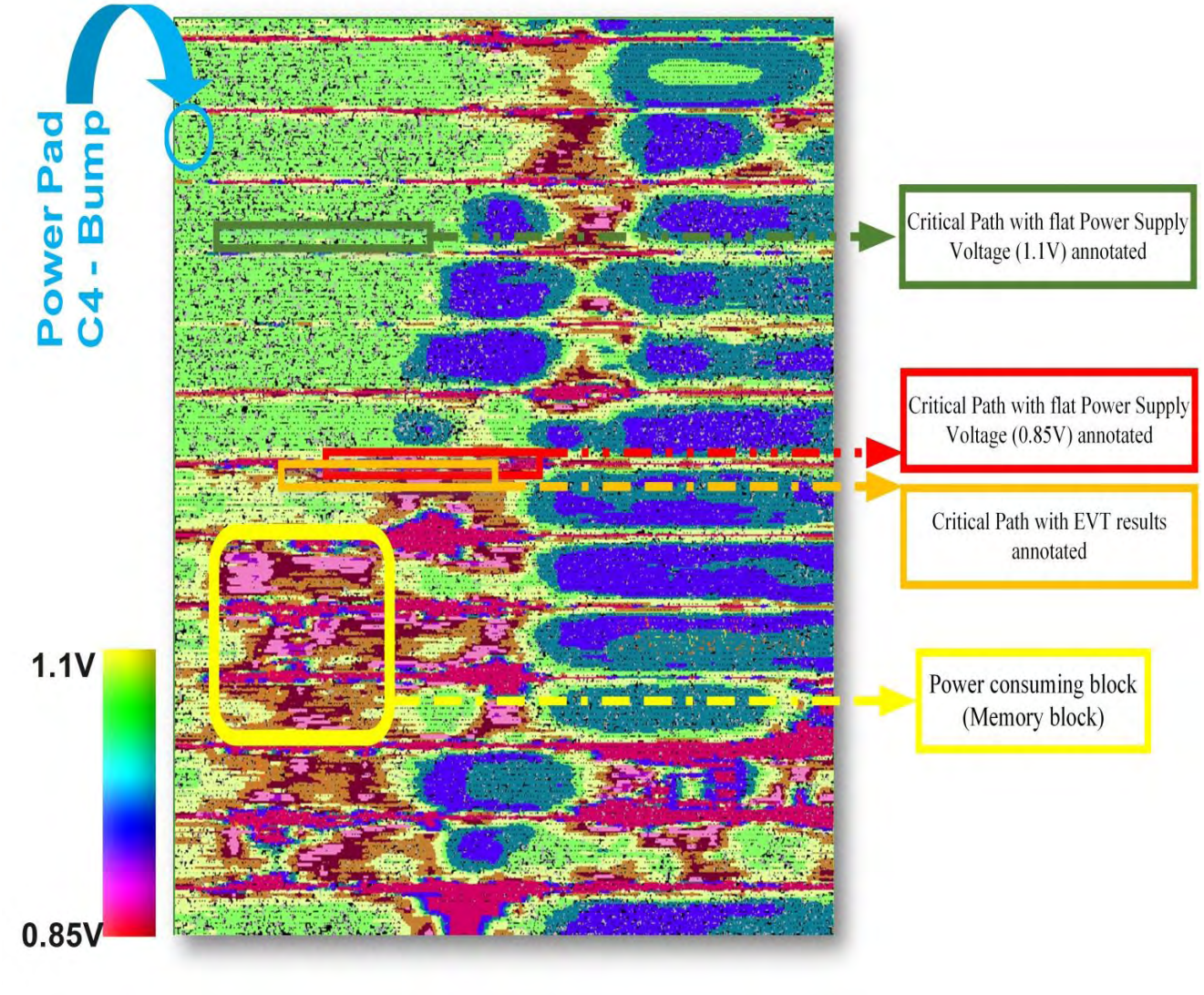


Figure 38 JPEG Voltage-drop colormap and location of Critical Paths according to different voltage levels annotation (Magma-Talus).

7.2 Voltage-Drop aware Static Timing Analysis

7.2.1 Related Work

Attempts to incorporate voltage drop [17] by annotating pre-calculated voltages (aiming to find a global time window of operation for each cell) on Static Timing Analysis or even attempts that consider Supply Voltage as a global variable [18], leading to better results, miss once again all the dynamically generated effects of the simulation and their interdependence, which are highly input pattern dependent.

7.2.2 Proposed Voltage-drop aware Static Timing Analysis

In modern IC designs, the lack of predictability of the worst case voltage level over each cell or block of the design during operation, lead designers to use flat in all IC cells the same worst case voltage level which is far more pessimistic than the real voltage level of the cell. This choice is made by designers in order to be on the safe side at the end of timing closure. This pessimistic and preservative approach result to over pessimistic assessment of the worst case delay and the structure of the critical path of the design during STA. The proposed approach makes use of the accurate results from the Statistical Prediction Engine of NANOPOWER™, which is based on Extreme Value Theory, for the accurate prediction of the worst case voltage over each cell of the design. As depicted in **Figure 39**, where V_{dd} is the nominal voltage and D_{vdd} the delay of each using nominal voltage level, the total delay resulting from STA (Synopsys PrimeTime®) using the EVT prediction for the worst case voltage on every cell of an IC is:

$$(7.1)\ D_{Total}^{EVT} = D_a^{EVT} + D_b^{EVT} + D_c^{EVT} + D_d^{EVT} + D_e^{EVT} + D_f^{EVT}$$

Following the preservative approach and annotating on every cell of a design a V_{worst}, say $0.8*V_{dd}$ the total delay is:

$$(7.2)\ D_{Total}^{Vworst} = D_a^{Vworst} + D_b^{Vworst} + D_c^{Vworst} + D_d^{Vworst} + D_e^{Vworst} + D_f^{Vworst}$$

The Critical Path is consists of the gates G_a to G_f. Since the gate delay when annotating V_{worst} will always be the same or worse than annotating EVT results ($D_a^{EVT} \leq D_a^{Vworst}$, $D_b^{EVT} \leq D_b^{Vworst}$, $D_c^{EVT} \leq D_c^{Vworst}$, $D_d^{EVT} \leq D_d^{Vworst}$, $D_e^{EVT} \leq D_e^{Vworst}$, $D_f^{EVT} \leq D_a^{Vworst}$) the total delay, being an aggregation of the individual gate delays, will always be worse than EVT prediction and thus :

$$(7.3)\ D_{Total}^{EVT} \leq D_{Total}^{Vworst}.$$

Other approaches trying to annotate, for the STA, different voltage levels on each cell, use directly the results from the dynamic simulation of the design. This approach is even worse than annotating flat in the design an extremely low-level voltage, since there is no guaranty that the annotated voltage levels on the cells are the real worst case voltage levels that these cells may “see” during operation.

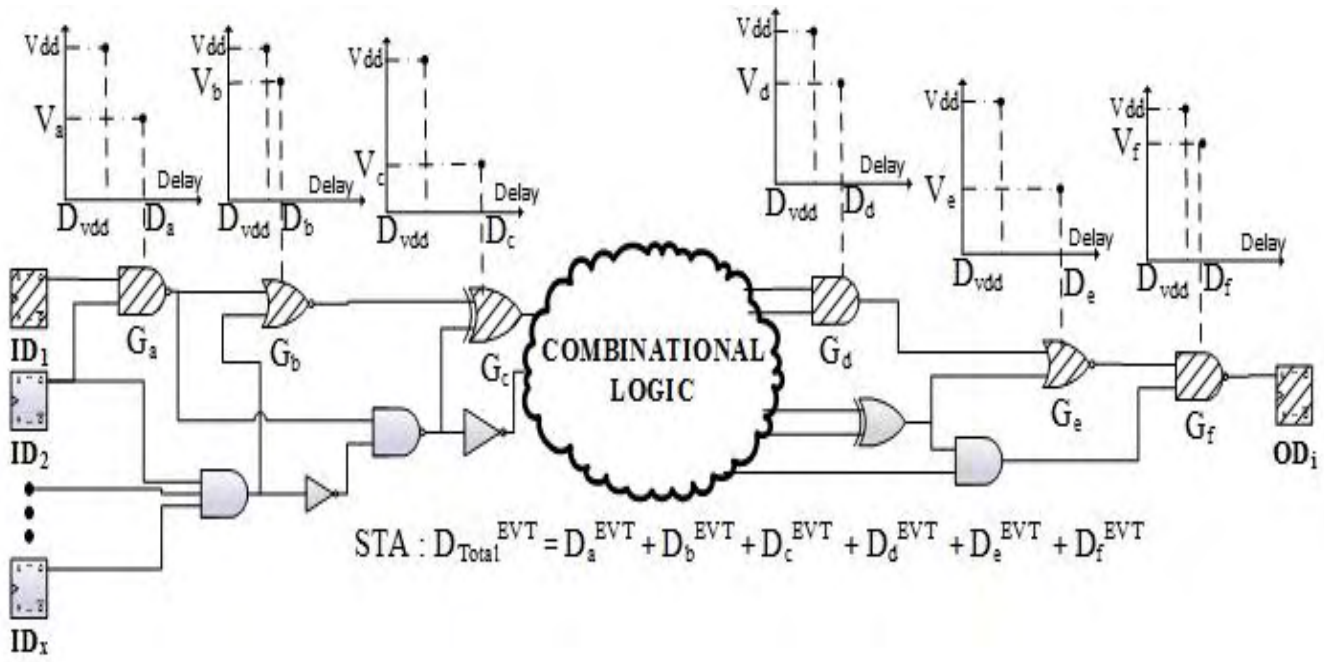


Figure 39 EVT results annotated on STA.

The results of the proposed methodology are depicted in **Table 4** for two industrial case designs and several ISCAS benchmarks. As it is obvious, the results of the proposed methodology, prove that using an over pessimistic voltage level during STA lead to unnecessary pessimistic delays and as we have already seen in the Section 7.1.2 also to wrong critical paths report. This in turn may lead designers to change gates or parts of the design that have no delay issues in reality and even worst to force a company (Design House) to an extra and unnecessary design cycle to fix problems that do not exist with a great impact on the Time-to-market for the design.

Table 4 Traditional STA vs EVT-STA.

IC	STA 1.1 V	STA 0.9 V	STA (EVT)	Margin (%)
H264	6.246 ns	7.196 ns	6.800 ns	5.50 %
JPEG	1.560 ns	1.834 ns	1.740 ns	5.12 %
C7552	0.480 ns	0.570 ns	0.520 ns	8.77 %
C6288	1.520 ns	1.800 ns	1.680 ns	9.33 %
C5315	0.580 ns	0.680 ns	0.660 ns	2.94 %
C3540	0.680 ns	0.800 ns	0.750 ns	6.25 %
C2670	0.570 ns	0.660 ns	0.630 ns	4.50 %
C1908	0.520 ns	0.600 ns	0.570 ns	5.00 %

7.3 Voltage-Drop aware Dynamic Timing Analysis

Traditional Static Timing Analysis (STA) techniques can take into account voltage drop in the form of specific values, which have been identified by the designers as the worst voltage level for each during IC operation. However, voltage drop effects are highly input pattern dependent. The lack of an accurate methodology for the estimation of the worst case voltage drop and its impact on IC's Timing, leads designers to be very pessimistic when incorporating the phenomenon during design closure. This over pessimism leads to the production of ICs adjusted to operate at a lower performance than they could actually reach. Looking for alternative approaches, one realizes that existing Dynamic Timing Analysis (DTA) techniques are based mainly on the simulation of input patterns, which are expected to produce the worst-case path delays. In this chapter we propose a novel methodology for voltage drop-aware Statistical Dynamic Timing Analysis, the results of which are less pessimistic and much closer to the real world than existing approaches.

7.3.1 Related Work

Existing methodologies in the area of Dynamic Timing Analysis use mainly pattern generation algorithms [13] [14] or path-based techniques [15] searching for a vector pair or appropriate input pattern that will generate the worst-case delay during simulation, trying to incorporate also the Voltage-drop effect. These approaches use restrictive assumptions in order to simplify the problem, such as calculating an effective Power/Ground net over the cells in order to estimate voltage drop and delay for each cell, building timing libraries. Thus the supply voltage is not considered as a global environmental variable, changing during the simulation. Graph-based techniques, using toggle rates and statistical methods in order to find the critical paths, [16] face similar problems [19] without being able to incorporate voltage-drop effect. We will try to briefly describe here the basic points of the most sophisticated from the existing approaches [14] for Dynamic Timing Analysis which considers voltage-drop effect.

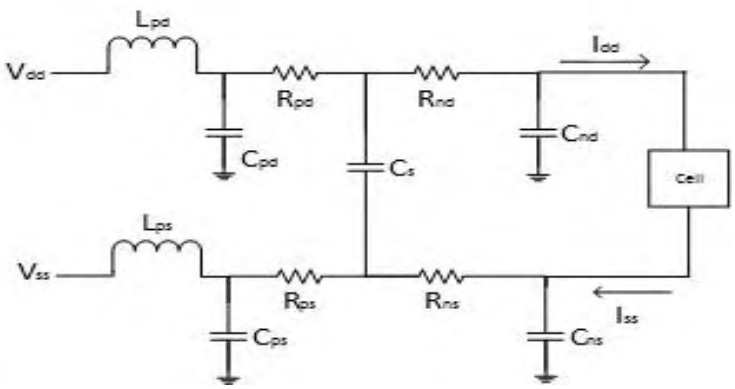


Figure 40 Circuit model for deriving supply noise

In **Figure 40** is depicted the Power and Ground Network used as the effective Power/Ground net used for performing a partial precharacterization of current and voltage waveforms for each cell. The proposed methodology is depicted in **Figure 41**. The method builds as a first step a library including all waveforms for all cells of the design. As a second step the method generates an initial set of input vectors that sensitize a path chosen by the designer, leaving as many Primary Inputs as possible to be

random. In the third step, the method performs iteration, limited to a pre-specified value, searching for input patterns causing worst case delay using the precharacterized libraries. At last step the resulted input patterns are used to perform Transistor level simulation of the design for more accurate estimation of the propagations delays.

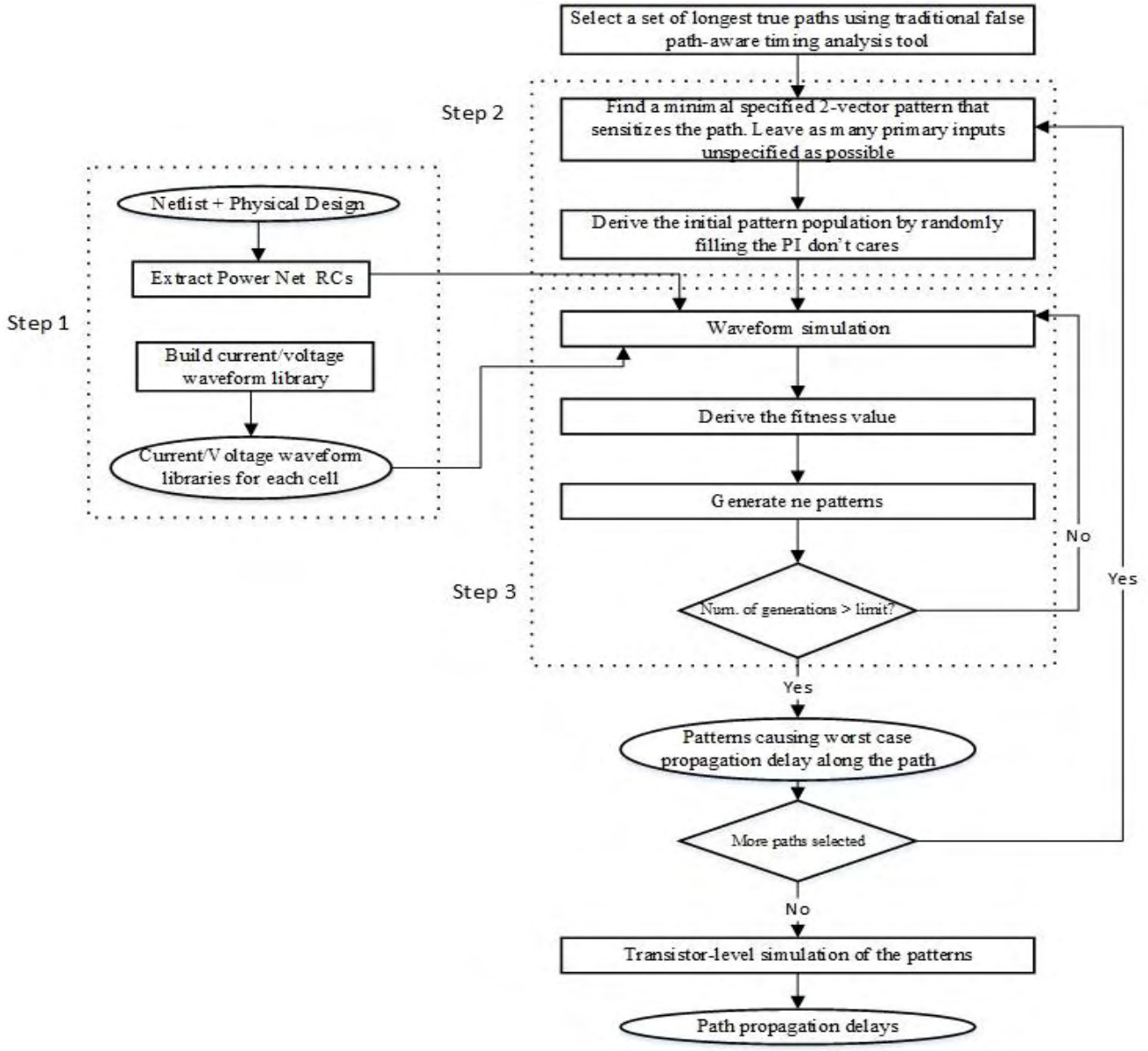


Figure 41 Pattern Generation taking into account the impact of the power supply noise on performance

As it is easily inferred the model used for emulating the voltage-drop effect in **Figure 41** is not accurate, missing the dynamic interdependent effects and also the algorithm used to find the one input pattern generating the worst case delay does not guaranty the quality of its results.

7.4 Proposed Voltage-Drop aware Statistical Dynamic Timing Analysis

In modern ICs which have a large number of primary inputs, say **n,** the number of all possible vector pairs is $\mathbf{4^n}$. Considering the complexity of modern ICs and the interdependent dynamic effects taking place during simulation, finding a vector pair or even a set of vectors, among $\mathbf{4^n}$ possible pairs, which will lead to the worst case delay, (slack) is practically impossible, constituting a problem which cannot be solved analytically. The proposed Statistical Dynamic Timing Analysis (SDTA) does not aim to find a specific vector pair or vector set that generates the worst case delay and also takes into account all the interdependencies induced by design complexity. Moreover considers the voltage drop as a global effect taking place during simulation. To achieve that, a powerful Statistical Prediction Engine was employed, using directly the timing results of a voltage drop-aware simulation. In both types of Dynamic Timing Analysis ("Direct" or "Statistical") the final result comes either directly as an outcome of the dynamic simulation of the design under test or after a statistical post-process step of these dynamic simulation outcomes. This means that the calculated or estimated (depending on the type of dynamic timing analysis) slack incorporates several dynamic interdependent effects which are present during the simulation of the design. Some of these effects are the voltage-drop and the thermal (increase of temperature within the DIE). Among the aforementioned effects, one which is dominant regarding the delay is the voltage drop over the power supply network of the design.

7.4.1 Identifying the Port of Interest

A preliminary step of the proposed Statistical Dynamic Timing Analysis is the identification of the output (ODi) of a path; usually a primary output or port of a sequential element of the design under consideration, where, designers would expect to see the maximum slack during dynamic simulation. The impact of Power Supply Noise (voltage-drop) over the cells of a design during simulation might be so large that could not just deteriorate the slack of the design but eventually change the expected critical path and thus the expected ODi experiencing the worst delay. To identify the real critical path and use its output port (*port of interest*) to perform the proposed SDTA; voltage-drop has to be incorporated into static timing analysis. In order to achieve this goal, we propose a three step procedure:

1. Perform a simulation on the IC under consideration, using random input vectors for the stimulation and then estimate for each individual cell (logic gate/block), the worst case voltage over its power supply pin (VDD-pin) during simulation [20].
2. Annotate the worst case voltages found in the previous step for each cell and run Static Timing Analysis (STA).
3. From all the reported critical paths choose the one (CPi) with the maximum slack (worst-case delay) and use the output port (ODi) of this path to drive the SDTA.

The impact of voltage drop on timing can be demonstrated through simple experiments. In Section 7.4.6 one may observe the differences on STA report, as a result of annotating different voltage levels at the VDD-pins of the cells of a design. The procedure for identifying the port of interest (ODi) is an auxiliary step in the proposed SDTA and thus is considered to be optional. Although this method recommends this step for a better and much closer to the real world choice of port ODi, the designer may use any other path output of his choice to perform the proposed SDTA.

7.4.2 Proposed ATPG

Having chosen the output port (ODi) for which we are interested in finding the worst slack (worst case delay of a logic state transition), either using the ODi which resulted from the proposed procedure or using another ODi, the first step of the proposed SDTA is the generation of the Input Pattern (Input Vectors – **Figure 42**). These Input Vectors will be used as stimuli for the voltage drop-aware dynamic simulation of the design under consideration and will ensure that a certain number of logic state transitions will be generated on ODi.

For the generation of these transitions, the isolation of all the cells that can possibly affect the logic state of ODi is necessary. The isolation step is performed by running a reverse BFS on the graph representing the design, starting from node ODi until it reaches all the nodes IDi, constituting input port (sequential element or primary input). The isolated cells along with ODi and all the IDi nodes (say ID1 to IDx) constitute a sub-design. The next step towards to the generation of the Input Pattern is the conversion of the sub-design to a logic function LFi expressing ODi in terms of ID1-IDx. LFi is then used as input to an SAT solver which runs twice iteratively.

The first time, SAT reports a set of 2500 different Bit Vectors (each Bit represents the logic value of the corresponding IDi) with values that force ODi to be in state of logic True and the second time reporting a set of 2500 different Bit Vectors that force ODi to be in state of logic False. These two runs are performed in parallel since no dependency exists between them. At each run, the algorithm calls SAT solver engine iteratively giving the previously reported solutions in order to be excluded from the current solution set. The iterations continue until the size of the solutions set (Bit Vectors) reach the number 2500. The aforementioned number of Bit Vectors is crucial for the Statistical Prediction Engine and it will be explained in the corresponding Section 7.4.4.

In order to create 2500 state transitions on ODi, we put interchangeably one Bit Vector from the solution set, which force ODi to be in logic state True followed by a Bit Vector from the solution set that force ODi to be in logic state False. The sequence of the Bit Vectors results from random choice. The outcome of this procedure is a set of Bit Vectors in a random sequence that toggles ODi from True to False and vice versa.

In order the voltage waveforms over the cells generated during the dynamic simulation of the IC, to be close to real world, the rest of the design should also be in operation mode. To achieve that the algorithm generates randomly complementary set of 2500 Bit Vectors at which, each Bit represents the logic value of an input port of the IC (sequential elements or primary inputs) except from ID1-IDx, in order to force the rest of the design to operate. The combination of the set including the Bit Vectors for ID_1-ID_x and the complementary set including the Bit Vectors for the rest input ports of the IC constitute a stimulus file with randomly generated Input Vectors. This file generates logic state transitions on OD_i, toggling a different set of cells, each time.

This procedure is important since the toggling on OD_i will finally generate the sample space of delays for the statistical analysis that follows. If the Input Vectors were generated without this consideration it would be practically not feasible to have the required logic state transitions on OD_i, using a rather small set of Input Vectors.

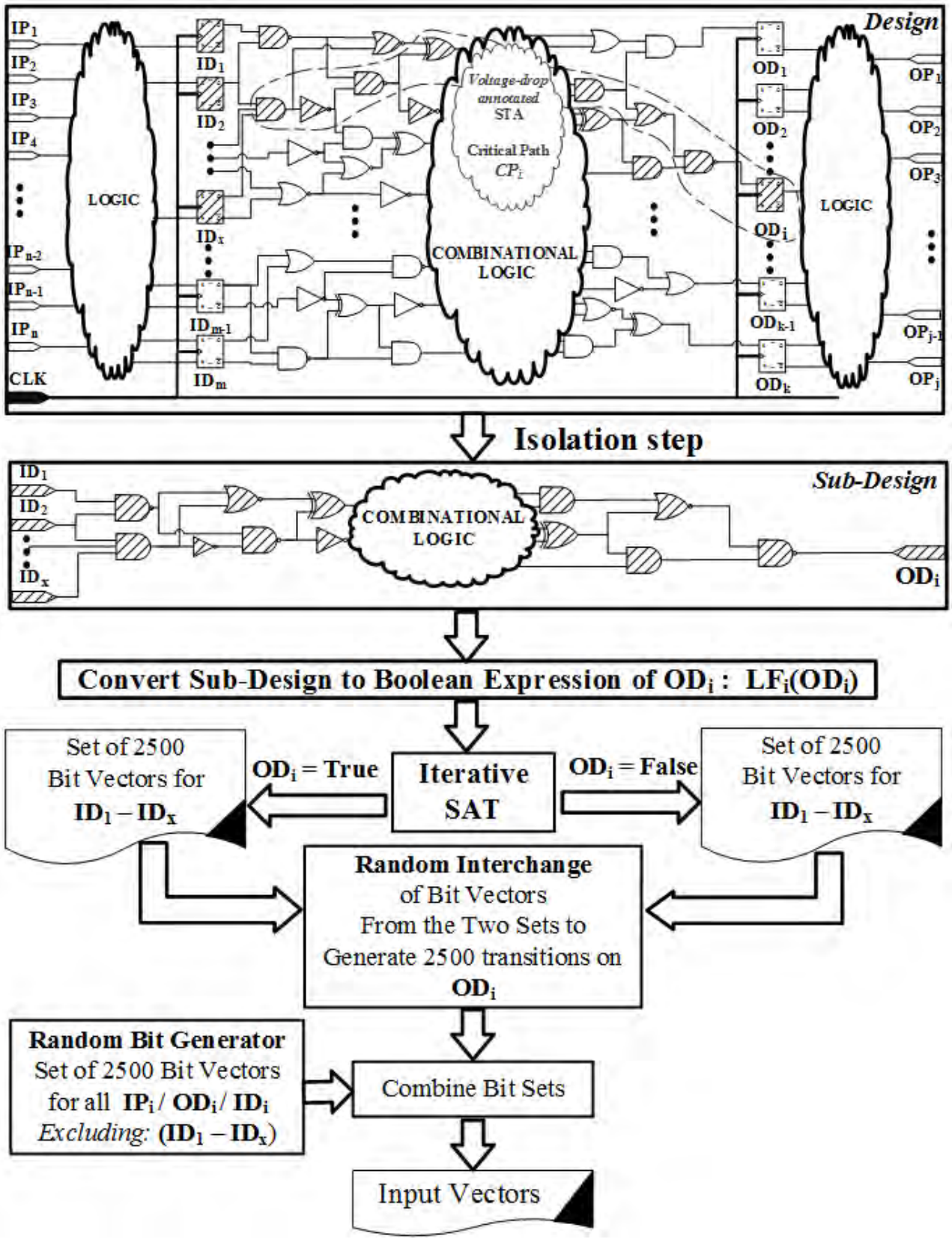


Figure 42 ATPG - Input Pattern Generation Process.

7.4.3 Delays Sample Space Generation

This section describes the voltage drop-aware dynamic simulation of the design under consideration using Input Vector files, the generation of which was described in the previous Section 7.4.2. The dynamic simulation aims to generate the necessary logic state transitions (call them logic events - E^i) and thus delays, at the output port OD_i under consideration. E^i results from the propagation, of all the events included in the Input Vector file (call it E^i_{VF}) for the corresponding simulation period, through the set of gates that are logically connected to OD_i (call it G_i set).

Each E^i arrives at OD_i on a specific time moment (call it D^i_m) within a simulation period *m* (of size T). Since the value of D^i_m is relative to the initial moment of the simulation period, it belongs to; we can call it delay of E^i. D^i_m results from the sum-up of the corresponding delays of all the individual events, generated during the propagation of E^i_{VF} through the gates included in set G^i (**Figure 43**). During a simulation period, the level of Power Supply Voltage on the power pin of a cell $\boldsymbol{G_h}$ may deteriorate significantly but the delay of the event $\boldsymbol{E^i_{Gw}}$ generated by $\boldsymbol{G_h}$ at its output port, is only affected by the voltage level on $\boldsymbol{G_h}$ power pin during the logic state transition that produces $\boldsymbol{E^i_{Gw}}$. This is the reason why annotating arbitrarily a static voltage level at the VDD-pin of the cells of an IC to perform STA, lead to overly pessimistic results.

The tool used for the dynamic simulation takes into account the interconnect parasitics between the cells and calculate the delay of each event using a set of precharacterized standard cell libraries, in several different voltage corners. As a result, the final event generated at OD_i for each period, incorporates also the delay induced by voltage drop. The corresponding delays-set from the set of events ($\boldsymbol{E_i}$) that were generated at OD_i during the dynamic simulation (using the random Input Vectors) will constitute a random sample of independent variables that will be used from the Statistical Engine.

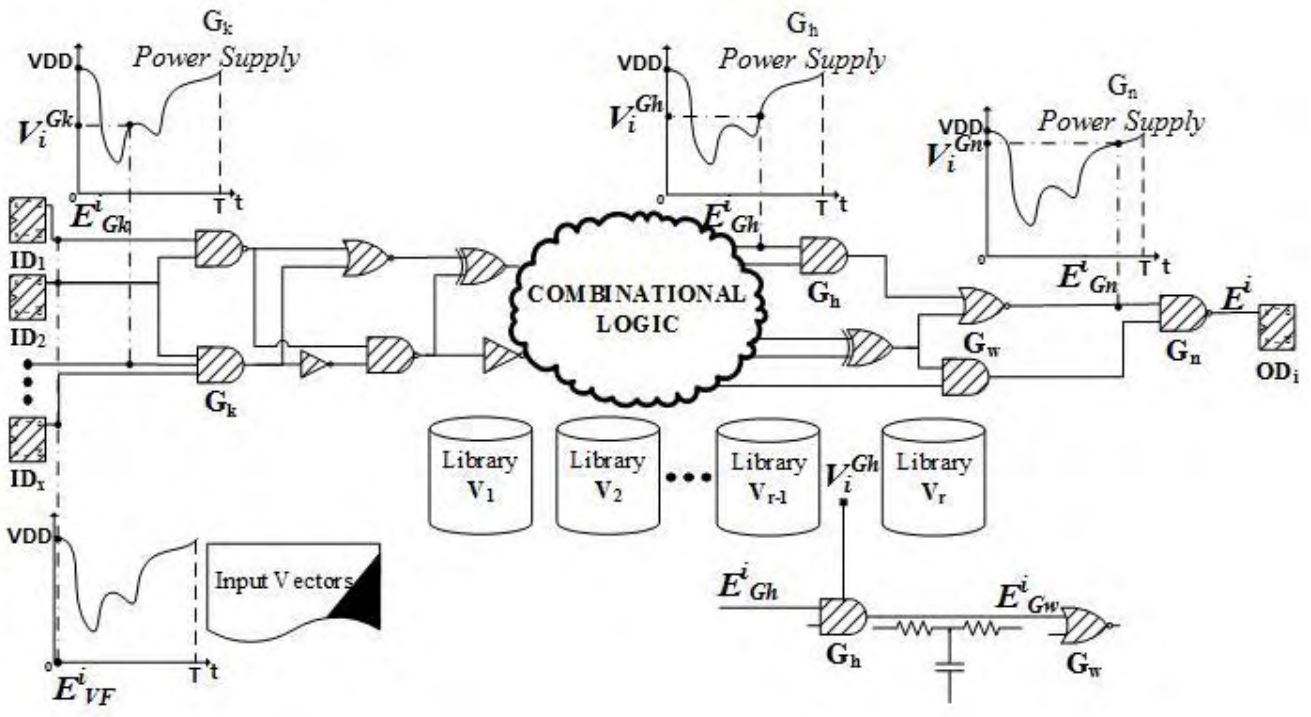


Figure 43 Supply Noise aware Dynamic Simulation.

7.4.4 Worst Case Delay Estimation by EVT

The bottleneck in obtaining the worst case delay $D_{max,i}$ over all possible delays that may occur at a chosen output port OD_i of a design is the identification of the exact input vector pair $(\underline{v_1}, \underline{v_2})$ which generates a state transition with the maximum latency among all the possible input vector pairs.

The individual delay D_i of each gate G_i involved in the logic state transition of OD_i is highly dependent on the Power Supply Noise over the gate G_i at the specific moment of the transition. The Power Supply Noise over G_i though, is a result of the switching activity, of all the gates and macro blocks existing in the design, during simulation.

From a statistical viewpoint, the entire set of input vector pairs that were generated for the simulation of the design under consideration can establish an initial population in which the cycle-accurate delay resulted on the port of interest OD_i is regarded as a random variable X. Supposing that X is characterized by a cumulative distribution function (cdf) $F(x)$ (and associated density function $f(x) = dF(x)/dx$), which is assumed to be *continuous* and *differentiable*, then the problem of determining the overall maximum port delay $D_{max,i}$ on OD_i can be cast as a problem of estimating the unknown maximum of a given statistical population with cdf $F(x)$. The latter quantity, known as the *upper* endpoint ω_F, is formally defined as the least upper bound of the range of values of the variable X (*supporting domain* or *support* of $F(x)$) [12]:

$$(7.4)\quad \omega_F = sup\{x \in \Re : 0 < F(x) < 1\}$$

which becomes $\omega_F = F^{-1}(1)$ if X is bounded from above or $\omega_F = +\infty$ in the opposite case. Let now $\boldsymbol{X} = [X_1, X_2, K, X_m]^T$ be a *random* sample of size m from the population at hand (i.e. the sample units X_i, $1 \le i \le m$, form themselves *independent* and *identically distributed* random variables with cdf equal to $F(x)$), and let $\boldsymbol{X}$ be partitioned into m/l sub-samples of size l from which the *maxima* units $Z_i = \max(X_{(j-1)l+1}, K, X_{jl})$, $1 \le j \le m/l$, are taken out to create a new sample $Z = \left[Z_1, Z_2, K, Z_{m/l}\right]^T$ of size m/l. Then an estimate for the upper endpoint ω of l can be computed as follows [22]:

$$(7.5)\quad \hat{\omega} = \hat{a} + \frac{\hat{b}}{1+l\sqrt{\pi \log l}\left(\mathrm{erf}\left(\sqrt{\log l}\right)-1\right)}$$

where $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$ is the well-known *error function* and $\hat{a}$, $\hat{b}$ are *maximum likelihood* (ML) estimates of parameters a, b that characterize the *asymptotic* distribution of the sample $\boldsymbol{Z}$ (provided that the size l of the sub-samples is sufficiently large). The latter is assumed to be the so-called *Gumbel* distribution which constitutes the *dominant* asymptotic distribution for maxima, as it was proved in [23] and validated experimentally in [22] (just as the normal distribution is actually the dominant - and not the only - asymptotic distribution of the central limit theorem inside a more general family known as *stable* [24]). This means that $\hat{a}$ and $\hat{b}$ are to be obtained by maximization of the following log-likelihood function of Gumbel distribution evaluated on the sample $\boldsymbol{Z}$:

$$(7.6)\quad \log L(a,b) = -\sum_{j=1}^{m/l}\left(\frac{z_j-a}{b} + exp\left(-\frac{z_j-a}{b}\right) + \log b\right)$$

A confidence interval (corresponding to a confidence level of $(1-\delta)\times 100\%$) has also been constructed in [22] for the estimate (7.5), and is given by:

$$(7.7)\quad |\widehat{\omega} - \omega| \le \frac{z_{\delta/2}}{\sqrt{m/l}}\frac{\hat{b}_n\sqrt{6}}{\pi} \cdot \sqrt{(\gamma-1)^2 + \frac{\pi^2}{6} + \frac{2(1-\gamma)}{1+l\sqrt{\pi\log l}\,(erf(\sqrt{\log l})-1)} + \frac{1}{\left(1+l\sqrt{\pi\log l}(erf(\sqrt{\log l})-1)\right)^2}}$$

where $Z_{\delta/2}$ is the $\delta/2$ quantile point of the standard normal distribution and $\gamma \approx 0.5772\ K$ is the *Euler gamma* constant [25].

7.4.5 Proposed Algorithm for SDTA

The implemented algorithm, of the proposed Statistical Dynamic Timing Analysis, is described by the following 7 steps:

1. *Identify the port of interest OD_i (sub-section 7.4.1) performing voltage drop-aware simulation and STA, annotating the reported voltage from the simulation.*
2. *Generate a total of $m = 2500$ random pairs $\{v_p, v_n\}$ of binary vectors for the circuit under consideration that result to 2500 state transitions at the output port OD_i of interest (sub-section 7.4.2). The selection $m = 2500$ for the number of input vector pairs (thus 2500 delay-samples) is explained below (last paragraph of this section).*
3. *Simulate the design (sub-section 7.4.3) under all generated vector pairs and record the delay on OD_i induced by the corresponding vector pair. The recorded data (delays) will constitute the random sample $\boldsymbol{X} = [X_1, X_2, K, X_m]^T$.*
4. *Arrange $\boldsymbol{X}$ into $m/l = 100$ sub-samples of size $l = 25$.*
5. *Construct the sample $\boldsymbol{Z}$ of the maxima units from the m/l sub-samples of $\boldsymbol{X}$.*
6. *Maximize the log-likelihood function (7.6) with respect to the parameters a, b, in order to obtain the ML estimates $\hat{a}$, $\hat{b}$.*
7. *Compute the upper endpoint estimate $\widehat{\omega}$ from (7.5) and - optionally - its confidence interval (for a given confidence level) from (7.7).*

Regarding Step 3 the computational time required to complete this step is entirely up to the simulator program employed, since there are many different simulators with speeds that range considerably depending on the detail of the analysis and their algorithmic efficiency. Although larger circuits will definitely take longer to simulate for every input vector pair, we must emphasize that a total of 2500 input vector pairs is sufficient to produce a reasonable EVT statistical estimate independently of the circuit size, as is further explained below.

Additionally in Step 4 the size l only needs to be adequate so that the sample of the maxima units from the sub-samples follows an asymptotic extreme value distribution. We have found by experimentation that $l = 25$ is a fair value. The number $m/l = 100$ of sub-samples (leading to a total of $m = 2500$ units) yields estimates with relative estimation error (i.e. quotient of confidence interval to estimate) of about 5% - at a confidence level 95% - for any delays set irrespective of its size or the size of the broader circuit, as was observed in [22]. This happens because with an increase in the delays set size, both the mean and the standard deviation of the

distribution of delays set are increased, but their ratio which determines the relative estimation error remains roughly constant. Only in the case where a smaller estimation error and/or a higher confidence level are desired, the number m/l of sub-samples will have to be increased (together with the total number m of input vector pairs). Step 6 is carried out via a standard unconstrained optimization algorithm (such as those described in [26]-[27]). Although this is essentially an iterative numerical procedure, the initial guesses of $\hat{a}$, $\hat{b}$ found by the method of moments [22] are extremely close to the final ML values and thus the optimization algorithm usually terminates in a matter of milli-seconds (and always converges to the global optimum).

7.4.6 Experimental Setup and Results

To demonstrate the benefits of the proposed voltage drop- aware Statistical Dynamic Timing Analysis (SDTA) we compare the results of the proposed method against the traditional Static Timing Analysis flow for the estimation of the worst case performance of the ICs. For our experiments we used two industrial ICs, the NOVA which is an H.264 baseline decoder and the MKJPEG which is a JPEG encoder, from the OpenCores benchmark suite along with six ISCAS benchmarks. All benchmarks were implemented in NanGate45nm technology process [28]. For the synthesis and place-and-route (P&R) flow, Synopsys IC Compiler® was employed. No optimizations for power and time were used during synthesis and P&R. For synthesis we used only simple logic standard cells (no macro blocks). In order to produce significant voltage drop over the Power Supply Network of the ICs, we used several metal layers for the generation of their Power Grid, using the smallest DRC (Design Rule Check) clean width on randomly chosen metal stripes of the Power Grid. For Power Supply we connected only one Power-Pad on their Power Grid. In **Table 5** is depicted, for each of the test ICs (*column I*) with size (*column II*), the runtime (*column III*) for the dynamic simulation, the nominal voltage of the process (*column IV*), the minimum voltage for each IC (*column V*) calculated at a cells' VDD-pin (not necessarily belonging to the sub-design involved in the toggling of OD_i) during the dynamic simulation, along with the corresponding voltage-drop (*column VI*) from the nominal voltage.

To identify the port of interest OD_i (Step 1) we employed NANOPOWER™ [20] [21], which is a SPICE-accurate voltage drop-aware simulator reporting the worst case voltage over each standard cell of a design during the simulation. We ran STA using Synopsys PrimeTime®. For the annotation of the worst case voltage at each standard cell, we used the command: 'set_voltage' (e.g.: *'set_voltage 0.921423 -cell FA_NOR3 -pg_pin_name VDD'*, [29] with the nominal voltage being 1.1V). Instead of NANOPOWER™ any other tool which performs voltage-drop analysis can be used and also instead of Synopsys PrimeTime® any other STA tool may be used too. The proposed SDTA used the resulted port OD_i from this step for all ICs in **Table 5**.

Table 5 Simulation Run-time and Worst Case Voltage-drop.

IC	IC (#cells)	Runtime (hours)	Nominal Voltage	Minimum Voltage	Voltage-drop (%)
H264	112935	16.213 hrs	1.1 V	0.9077 V	17.473 %
JPEG	29674	3.620 hrs	1.1 V	0.8949 V	18.642 %
C7552	6759	0.970 hrs	1.1 V	0.8894 V	19.145 %
C6288	3475	0.461 hrs	1.1 V	0.8789 V	20.100 %
C5315	3253	0.468 hrs	1.1 V	0.9051 V	17.718 %
C3540	2902	0.416 hrs	1.1 V	0.9121 V	17.081 %
C2670	1710	0.245 hrs	1.1 V	0.9582 V	12.890 %
C1908	819	0.110 hrs	1.1 V	0.9611 V	12.627 %

For the generation of the random input pattern (Step 2), we developed a program in Python, using the core engine of the Z3 SAT-solver [30]. The results from this step were verified using Synopsys VCS®, ensuring that the generated input pattern will produce the proper number of logic state transitions on the port of interest (OD_i). The resulted Input Vectors file is exported in VCD format (Value Change Dump). The part of this step that generates the two sub-sets of 2500 different input vectors that set OD_i to logic True and False correspondingly can be performed using an Automatic Test Pattern Generator or by making use of other SAT solver.

The tool employed for the voltage drop-aware dynamic simulation (Step 3), generating the sample space of delays, was NANOPOWER™. The tool simulate each IC using the corresponding input pattern from Step 2 and reports all the events generated on OD_i, during the Power Supply Noise aware dynamic simulation, along with their corresponding delays in a file (OD_i_name.drpt).

To perform the statistical analysis (Steps 4-7) based on Extreme Value Theory (EVT) we developed a tool (Statistical Prediction Engine) in C++ code which takes as input the delay report file (OD_i_name.drpt) from the dynamic simulation. The result of the Statistical Prediction Engine is the worst case delay of all the possible logic state transitions that may be generated at the port of interest OD_i during dynamic simulation. The Statistical Engine optionally exports the corresponding confidence-interval.

The comparison between the proposed Statistical Dynamic Timing Analysis and the traditional STA methodology is demonstrated in Table II where *column II* presents the results of the STA after annotating at all cells the nominal voltage of the technology. *Column III* presents the results of STA after annotating at all cells 0.9V (~18.2% voltage drop). In *column IV (SDTA)* we demonstrate the results of the proposed Statistical Dynamic Timing Analysis, the runtime of which is negligible (less than 1min for the 2500 samples used). *Column V* presents the existing delay margin between *column III* and *column IV*. To test the quality of the results for the proposed SDTA, we ran Monte Carlo analysis for the small benchmarks, where it is feasible. The results are demonstrated in the last column, *column VI* which presents the worst out of all the delays (for all output ports, not just OD_i) reported by the voltage drop-aware dynamic simulation for 100K random vectors. As it is obvious no reported delay, from the100K simulations, exceeds the EVT prediction of the worst case delay for all ICs.

Table 6 Traditional STA vs SDTA vs DTA-Monte Carlo.

IC	STA 1.1 V	STA 0.9 V	SDTA (EVT)	Margin (%)	DTA Monte Carlo
H264	6.246 ns	7.196 ns	6.592 ns	8.39 %	-
JPEG	1.560 ns	1.834 ns	1.691 ns	7.80 %	-
C7552	0.480 ns	0.570 ns	0.500 ns	12.28 %	0.49 ns
C6288	1.520 ns	1.800 ns	1.600 ns	9.44 %	1.58 ns
C5315	0.580 ns	0.680 ns	0.620 ns	7.35 %	0.61 ns
C3540	0.680 ns	0.800 ns	0.730 ns	8.75 %	0.71 ns
C2670	0.570 ns	0.660 ns	0.600 ns	6.06 %	0.59 ns
C1908	0.520 ns	0.600 ns	0.570 ns	5.00 %	0.57 ns

From the demonstrated results we can conclude that the existing Timing Analysis flow is overly pessimistic, while the proposed SDTA can safely predict the worst case delay on a chosen design port, taking into account the voltage drop and the interconnect parasitic. The accuracy of the method was verified by the findings presented in *column VI* of **Table 6**. To find the slack and power information at specific voltage level, in order to perform the STA runs and the dynamic simulations, we prepare Synopsys Liberty (.lib) files at several voltage corners from 1.1V to 0.8V in 0.05V increments, using the precharacterization tool Silicon Smart from Synopsys®.

The proposed SDTA uses industry standard file formats and it is easily integrated with existing industry EDA tools.

8 Power and Timing Joint Process Variation

8.1 Introduction

During IC tape-out, several transistor attributes vary due to the lack of precision in the fabrication process. Failure of the repeatability of the transistor pattern for each process node, results in the process variation effect. Process variation introduces also variation on the performance of the IC, exceeding this way the specification on the measured results. The main attributes, of the transistors, which are mostly affected by process variation, are the channel length, the width and the thickness of the oxide. The main impact of the effect is the variation on the voltage threshold (VTH) of transistors. Especially in process nodes lower than 60nm the impact of the effect becomes dominant as the variation becomes an even larger percentage of the nominal length and width size of the devices since the last, have become so small that are approaching the fundamental dimensions size of atoms and the wavelength of usable light for patterning lithography masks. The process variation, as it is easily deduced, plays a major role on the overall yield of the fabricated IC.

From the aforementioned description it is clear that Process Variation has to be taken into account during the design cycle of the IC design. Nowadays all chip manufacturing companies model the variation at each process node and integrates them along with the rest of the PDK data. Considering the nominal (default) sizes of the transistor attributes to be the target, the variation of each size most often is modeled as a Gaussian distribution of the size with mean value the nominal value of each size and the σ is induced from the corner models of the process. As it has been described in all previous Chapters of this thesis the key performance metrics of an IC are the results of power and timing analysis. In other words the process variation data are used by the design houses to predict power consumption and delay of the IC.

Existing methodologies on process variation, try to incorporate the effect separately on power and timing analysis. Moreover the variation on transistor sizes is incorporated in the final results through additional statistical analysis, like Monte Carlo. These approaches are considered to be accurate enough until the past years. The statistical analysis performed though is very simple and does not report the worst case prediction based on the simulations incorporating variation. During the previous chapters it has been established the importance of the existing correlation between power and timing. None of the existing methodologies in process variation is in a position to introduce an approach which will be able to handle the two types of simulation simultaneously. Thus the main question whether there is a method to estimate the probability, the performance of an IC to exceed a specification limits in either Power or Delay (or both), remains unanswered.

To be able to tackle the problem of predicting the worst case performance of an IC due to process variation joining the two key performance goals, power and timing, we introduce a novel approach based on Multivariate Extreme Value Theory. The methodology consists of the following steps:

1. Generate a set of 2500 random samples for Length.
2. Generate a set of 2500 random samples for VTH.
3. Perform STA on the IC using the paired values from the “Length” sample space.
4. Perform simulation on the IC using the paired values from the “VTH” sample space.

5. Form of bivariate Extreme Value Distribution:
    - F1(x1),F2(x2): univariate EV marginal distributions
    - l(y1,y2): EV dependence structure
6. Bivariate EV Estimation:
    - Perform univariate EV estimation for Delay and Power (via ***maximum likelihood***)
    - Estimate dependence structure (via **censored likelihood**)

The result of the aforementioned methodology is a surface graph describing the probability for the IC under test to exceed jointly the worst-case power consumption and delay.

8.2 Approximation of the tail of a probability distribution via EVT

In this section we describe a procedure for estimating a very low probability in the right tail of a distribution via EVT. Let X is a random variable of delay or power and F(x) = Pr[X<x] is its (cumulative) distribution function (df). Let also $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ be a random sample of values of $\boldsymbol{X}$ obtained by varying a statistical population of process parameters (for example, transistor length L and threshold voltage VTH). By "random" here we mean that the units $X_1, X_2, \ldots, X_n$ constitute independent and identically distributed (iid) random variables with df F(x) each. The units $X_1, X_2, \ldots, X_n$ of $\boldsymbol{X}$ can be sorted in ascending order as $X_{1:n} \leq X_{2:n} \leq \cdots \leq X_{n:n}$ and the *i*-th unit $X_{i:n}$ of this sequence is called the *i*-th *order statistic* of $\boldsymbol{X}$. Let us suppose that the *k* upper order statistics $X_{n-k+1:n}, X_{n-k+2:n}, \ldots, X_{n:n}$ belong to the right tail of the distribution. In other words, let the $(n-k)$-th order statistic $X_{n-k:n}$ constitute a high threshold *u* marking the beginning of the right tail. The sample $X_{ex} = (X_{n-k+1:n}, X_{n-k+2:n}, \ldots, X_{n:n})$ of the order statistics which are larger than $u \equiv X_{n-k:n}$ is called the sample of *exceedances* over threshold *u*. It is not difficult to infer that this sample follows a distribution $F_u(x)$:

$$(8.1) \qquad F_u(x) = \Pr[X \leq x \mid X > u] = \frac{\Pr[X \leq x, X > u]}{\Pr[X > u]} = \frac{F(x) - F(u)}{1 - F(u)}, \qquad x \geq u$$

Now, a fundamental result from EVT states that *irrespective* of $F(x)$ and under conditions normally satisfied in practice, the exceedance df $F_u(x)$ approaches asymptotically (i.e. for large enough *u*) a *generalized Pareto* (GP) distribution $G_{\beta,\gamma}(x)$ with the following form:

$$(8.2) \qquad G_{\beta,\gamma}(x) = 1 - \left(1 - \gamma \frac{x-u}{\beta}\right)^{1/\gamma}, \quad \begin{array}{ll} u \leq x \leq +\infty & \text{if} \quad \gamma \leq 0 \\ u \leq x < u + \beta/\gamma & \text{if} \quad \gamma > 0 \end{array}$$

Where $\beta > 0$ and $\gamma \in \mathbb{R}$ are parameters and $u \equiv X_{n-k:n}$ is the chosen constant threshold. The parameter β is a *scale* parameter that characterizes the spread of the distribution around its mean, and roughly corresponds to the standard deviation parameter σ of the normal distribution. The parameter γ is called *shape* parameter or *tail index*, and is more important since it is connected with the right tail of the parent distribution $F(x)$. If $\gamma < 0$ then $F(x)$ has a long (or *heavy*) right tail and the associated random variable *X* is unbounded from above (i.e. $sup\{x: F(x) < 1\} = +\infty$). If $\gamma > 0$ then $F(x)$ has a short right tail and the random variable *X* is upper bounded (i.e. $sup\{x: F(x) < 1\}$ is finite). The case $\gamma = 0$ is interpreted as $\gamma \longrightarrow 0$ [reducing the GP to the *exponential* distribution] $G_{\beta,0}(x) = 1 - \exp(-x/\beta)$ and indicates that $F(x)$ has a moderate right tail.

If we can fit the GP distribution (8.2) to the sample of exceedances $\boldsymbol{X_{ex}}$ (i.e. select appropriate values $\hat{\beta}$ and $\hat{\gamma}$ of the parameters β and γ) so as to approximate $F_u(x)$ by $G_{\hat{\beta},\hat{\gamma}}(x)$, and also approximate $1 - F(u) = 1 - F(X_{n-k:n}) = \mathbf{Pr}[X > X_{n-k:n}]$ by the percentage k/n of upper order statistics in total sample size, we can estimate the unknown $F(x)$ for $x \geq u$ (i.e. for points further into the tail) from (8.1) and (8.2) as:

$$(8.3) \qquad F(x) = (1 - F(u))F_u(x) + F(u) = 1 - (1 - F(u))(1 - F_u(x)) \approx 1 - \frac{k}{n}\left(1 - \hat{\gamma}\frac{x - X_{n-k:n}}{\hat{\beta}}\right)^{1/\hat{\gamma}}$$

The standard way of deriving estimates of the unknown parameters of a distribution on the basis of a sample is via *maximum likelihood* (ML). ML amounts to maximizing the joint density function of the iid units of the

sample w.r.t. the unknown parameters (i.e. the *likelihood* function), or more typically its natural logarithm (known as the *log-likelihood* function). The density function of the GP distribution (8.2) is:

$$(8.5)\qquad g_{\beta,\gamma}(x)=\frac{dG_{\beta,\gamma}(x)}{dx}=\frac{1}{\beta}\left(1-\gamma\frac{x-u}{\beta}\right)^{(1/\gamma)-1},\quad \begin{array}{ll} u\le x\le+\infty & \text{if}\quad \gamma\le 0\\ u\le x<u+\beta/\gamma & \text{if}\quad \gamma>0\end{array}$$

and the corresponding log-likelihood function for the sample $\boldsymbol{X_{ex}}$ is:

$$(8.6)\qquad \log L(\beta,\gamma)=\log\prod_{i=1}^{k}\left(\frac{1}{\beta}\left(1-\gamma\frac{X_{n-i+1:n}-X_{n-k:n}}{\beta}\right)^{(1/\gamma)-1}\right)=-k\log\beta+\left(\frac{1}{\gamma}-1\right)\sum_{i=1}^{k}\log\left(1-\gamma\frac{X_{n-i+1:n}-X_{n-k:n}}{\beta}\right)$$

The maximization of the above bi-variate function w.r.t. (β,γ) to obtain the ML estimates $(\widehat{\beta},\widehat{\gamma})$ can be performed by numerical routines for multivariable nonlinear optimization. More effectively, we can perform a reparameterization $(\beta,\gamma)\longrightarrow(\boldsymbol{\tau},\gamma)$ with $\boldsymbol{\tau}\equiv\gamma/\beta$, and differentiate (8.6) w.r.t. $\boldsymbol{\tau}$ and γ. The estimate $\hat{\boldsymbol{\tau}}$ can then be found via numerical solution of the following one-dimensional equation in the interval $\boldsymbol{\tau}\in(-\infty,(X_{n:n}-u)^{-1})$:

$$(8.7)\qquad \frac{1}{\tau}+\left(\frac{1}{\sum_{i=1}^{k}\log\left(1-\tau(X_{n-i+1:n}-X_{n-k:n})\right)}+\frac{1}{k}\right)\sum_{i=1}^{k}\frac{X_{n-i+1:n}-X_{n-k:n}}{1-\tau(X_{n-i+1:n}-X_{n-k:n})}=0$$

and the estimate $\hat{\gamma}$ (and $\hat{\beta}=\hat{\gamma}/\hat{\boldsymbol{\tau}}$) can be computed by:

$$(8.8)\qquad \hat{\gamma}=-\frac{1}{k}\sum_{i=1}^{k}\log\left(1-\hat{\tau}(X_{n-i+1:n}-X_{n-k:n})\right)$$

An important question in tail estimation via EVT concerns the appropriate number *k* of upper order statistics that are assumed to belong to the tail. In determining *k* we are faced with a tradeoff between variance and bias. ML estimation of β,γ on the sample $\boldsymbol{X_{ex}}$ requires *k* to be large so that the estimates $\widehat{\beta},\widehat{\gamma}$ are accurate, i.e. they have small variance. A too large *k*, however, may incorporate units that do not belong to the tail, and therefore cause $\boldsymbol{X_{ex}}$ to deviate from its asymptotic GP distribution, i.e. introduce bias. A rule of thumb that is most often used in practice is to set *k* at the upper 10% of the sample **X**, i.e. fix $k=0.1n$.

8.3 Multivariate Extreme Value Theory

Many problems involving extreme events are inherently multivariate and require statistical methods for analyzing extremes of multivariate data. In particular, given a sample of multivariate observations which is assumed to form independent and identically distributed random vectors, multivariate EVT attempts to extrapolate outside the range of the available data. This amounts to estimating the *tail* of the underlying multivariate distribution, i.e. estimate with good accuracy the probability of an event to lie in a region of the sample space with none or very few of existing observations. The study of multivariate extremes splits apart into two components: the *marginal* distributions and the *dependence* structure. The first step merely involves the univariate techniques described in the previous section, while the second step entails the construction of sensible parametric models for the multivariate dependence function. The interested reader can find comprehensive coverage of multivariate extreme value theory in, for instance, Galambos (1978, 1987) [12], Resnick (1987) [23], Beirlant, Goegebeur, Teugels and Segers (2004) [31], de Haan and Ferreira (2006) [32], Falk, Husler and Reiss (2011) [33] , Kotz and Nadarajah (2000) [34] , Coles (2001) [35] , Drees (2001) [36] , Reiss and Thomas (2001) [37], Fougères (2004) [38].

Historically, the first statistical methods for multivariate extremes follow the approach of sample maxima. The approach consists of partitioning a sample of multivariate observations into blocks, each typically corresponding to one interval of observations, and fitting a multivariate extreme value distribution to the sample of component-wise block maxima. However, multivariate block maxima almost never occur simultaneously, and also reducing the sample to a single observation per period disregards the possibility of witnessing several relevant events in the same period. More efficient is to use the *excesses over a high multivariate threshold* approach, i.e. all observations for which at least one coordinate exceeds a high threshold (and which can be considered as lying on the tail of the underlying multivariate distribution).

Specifically, let $F(x_1, x_2, \ldots, x_d) \equiv F(\mathbf{x})$ be a *d*-variate distribution function and let $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$ be a sample of independent and identically distributed *d*-dimensional random vectors, each with df $F(\mathbf{x})$. As in the univariate case, a fundamental result from EVT states that *irrespective* of $F(\mathbf{x})$ and under conditions normally satisfied in practice, the *tail* of $F(\mathbf{x})$ can be approximated by the following *d*-variate extreme value distribution:

(8.9) $$G(\mathbf{x}) = \exp\left(-l\left(-\log G_1(x_1), -\log G_2(x_2), \ldots, -\log G_d(x_d)\right)\right)$$

where $G_1(x_1)$, $G_2(x_2)$, … , $G_d(x_d)$ are univariate marginal extreme value distributions and $l(y_1, y_2, \ldots, y_d)$ is the multivariate dependence function.

The univariate extreme value marginals $G_j(x_j)$, $j = 1, \ldots, d$ have the parametric form (8.3), and are obtained by the generalized Pareto (GP) distribution (8.2) and the fact that the $(n-k)$-th order statistic $X_{n-k:n}$ of each univariate sample (considered as a high threshold u_j $j = 1, \ldots, d$) lies in the tail of the corresponding univariate marginal.

The most popular model for dependence function is the *logistic* model:

(8.10) $l(y_1, y_2, \ldots, y_d) = (y_1^{1/\alpha} + y_2^{1/\alpha} + \cdots + y_d^{1/\alpha})^{\alpha}, \qquad 0 < \alpha \leq 1$

which was introduced by Gumbel [39] and has been widely used ever since due to its tractability and its applicability in a broad range of practical cases [40]. This is a parametric model with parameter α which determines the degree of dependence between the components of the multivariate distribution. In particular dependence weakens as α increase, with the two limiting cases $\alpha \downarrow 0$ and $\alpha = 1$ corresponding to complete dependence and independence respectively.

The parameters β_j and γ_j, $j = 1, \ldots, d$ of the univariate extreme value marginals can be estimated on the basis of the acquired sample by maximum likelihood via (8.7) and (8.8), while the dependence parameter α of the logistic model can also be estimated by maximum likelihood, leading to the overall procedure of *censored likelihood*. The details can be found in [41].

In the particular case of variation of delay and power with process parameters, some examples of the univariate marginal distributions along with their bivariate joint distribution are shown in the following figures:

8.4 Experimental Setup and Results

In order to verify the proposed methodology an experiment has been set up using NanGate45nm process node. Using the typical values for length and VTH from the PDK and 3σ the value of the sizes from the fast and slow corner models we generated a samples space (2500 samples) forming a Gaussian distribution for each size. Then using each pair of values we ran STA (Synopsys NanoTime) changing the length and VTH on the fly for each run through a custom made script. The runs were parallel and the results form a Gaussian distribution also as depicted in **Figure 44**.

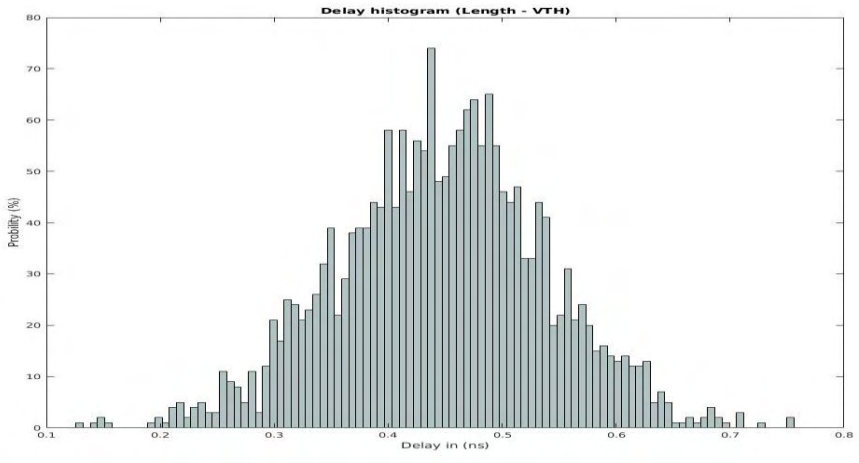


Figure 44 Delay histogram (Synopsys NanoTime).

Then for the same pairs of length and VTH we ran random vector simulation (Synopsys FineSim) reporting the average and peak power for the simulated vectors. The results also form a Gaussian distribution.

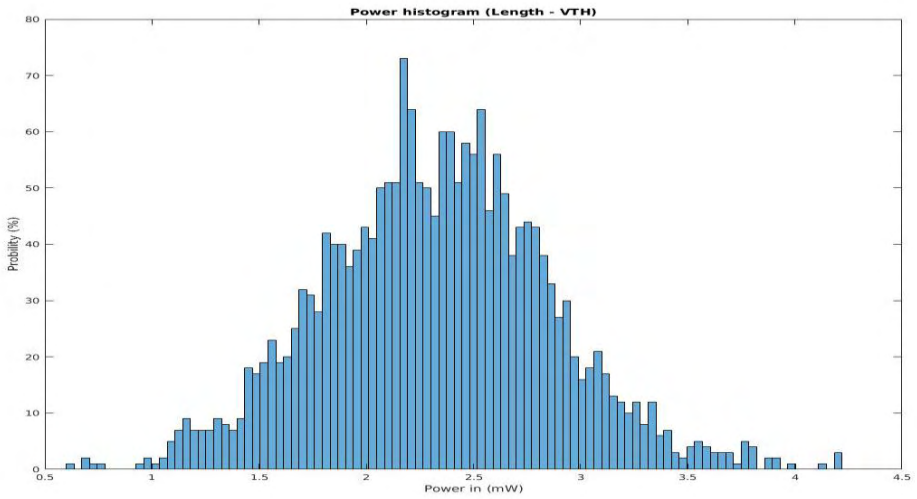


Figure 45 Power histogram (Synopsys Finesim).

Finally following the steps of the proposed methodology we end up to a surface (**Figure 46**) where the X and Y axis are the extreme power and timing reports and the Z axis is the joint probability to simultaneously the IC exceed delay and power specification.

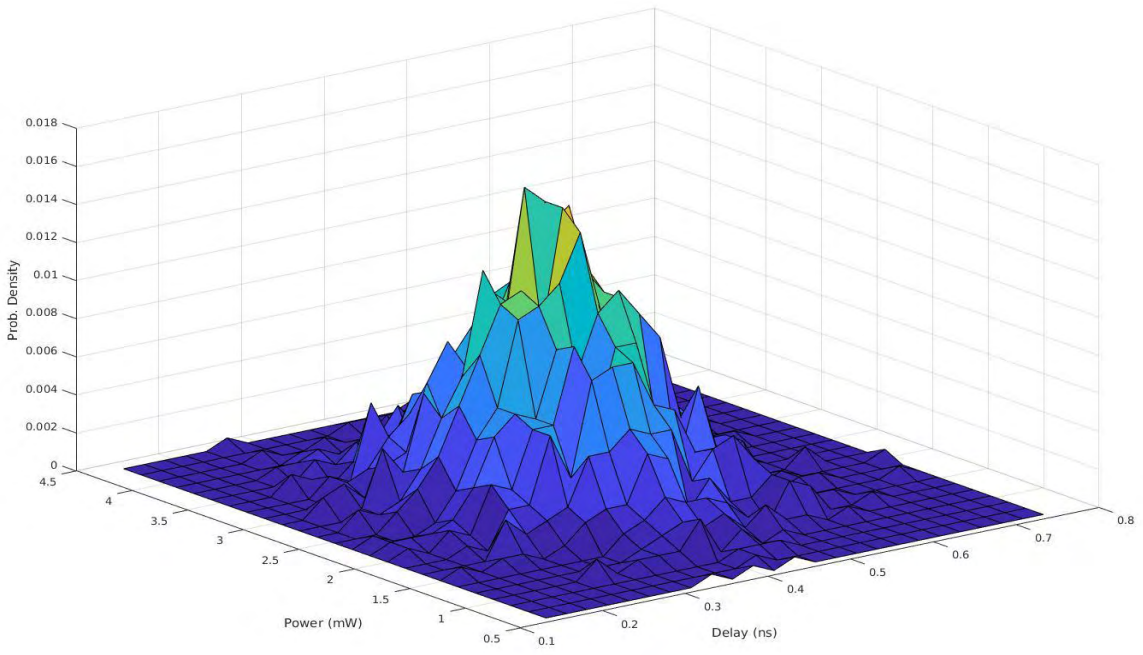


Figure 46 Surface - Joint Delay & Power process variation.

9 High Frequency Clock Trees in modern ICs

Rising operating frequencies in modern deep sub-micron ICs lead to faster clocks, which in turn means smaller clock period. The clock pulse though is a pulse wave. From the aforementioned we can easily deduce that as frequency increases the wavelength becomes smaller and smaller. On the other hand modern ICs are getting larger and thus all the interconnects become longer. The decrease of the pulse width along with the simultaneous increase of the size of the interconnects make these two size to approach each other. When the wave-length is able to fit within the length of the interconnects then the last, do not behave as lumped elements but as distributed and they cannot be modeled using RC-based models (R which is resistance, L which is inductance, C which is capacitance or G which is the conductance). In this type of operation wires are modelled as transmission lines using usually a lumped model like the one in **Figure 48**. The R and G are used to model the losses of energy in transmission lines while L and C are used to model the magnetic and electric energy storages.

Traditional clock networks used inverter and buffer chains, which are not energy-efficient at the multi-GHz clock rate since the charging/discharging power consumption of a wire is $C_w V_{DD}^2 f$, which cannot be improved by the Dennard's scaling down process; C_w remains almost constant for a fixed length wire, V_{DD} does not scale down and consequently f is still limited by the power budget. In addition, signal integrity issues caused by the buffers limit the performance of this approach. To alleviate these problems, multiple alternative methods have been proposed to reduce the capacitance load, using transmission lines instead of the wires, and also distributing the whole clock generators over the chip. The basic alternatives for the design of the clock distribution network are current mode logic (CML) clocking, capacitively driven wires (CDW), LC-resonance, and travelling and standing wave schemes using transmission lines. Out of these alternatives the transmission line based clock distribution methods, proved to be the most suitable to lower jitter and skew for high-speed clock signals in addition to the power efficiency. Indicatively transmission lines compared to traditional buffer chains methodologies when distributing a 2.5GHz clock over a 5mm distance in a 90nm process, proved to have ~200 times lower jitter, ~7.5 times less skew and ~2.9 times less power.

The power consumptions of clock networks, could be derived from $CV^2 f$, where C is the total capacitance of the wiring and loads. As the frequency increases, the power linearly increases as well. We can use the transmission lines (TL) instead of narrow wires to propagate the clock over the chip; the power can be calculated from driving a Z_0. Load power in addition to the TL losses (RI^2 where R is the loss). Generally there are two approaches for implementing this: A. Travelling wave, B. Standing wave. In the travelling wave approach, a closed loop path (differential or single) with inverting stages is used in order to generate the oscillating signal over the chip. These clock sources are called rotary travelling wave oscillators (RTWO) and the clock phase at loads depend on the position of loading nodes in the loop. Transmission lines can be implemented in the top thick metal layers with low sheet resistances. Since they are passive elements, they are less sensitive to process variation and do not add up to the jitter noise. In some applications of the distributed networks, designers used PLL (Phase Locked Loop) or DLL (Delay Locked Loop) blocks to control the skew and jitter of the clock.

9.1 Transmission Lines

Coplanar line geometries are widely adopted in RF and mmWave IC designs due to several reasons. Coplanar transmission lines are very friendly when it comes to characterization with on-wafer measurements, since the typical RF and mmWave measurement probes are configured with Ground-Signal-Ground (GSG) probe tips. The symmetrical ground planes in CPW-type transmission lines provide good isolation to surrounding electromagnetic aggressors and at the same time allow for easy shunt connections to ground of other circuit devices. A variety of coplanar transmission lines such as CPW, CPWG, and SCPW are investigated both in theory and by silicon experiments.

In general the SCPW transmission line seems to outperform the regular CPW in terms of loss with the additional advantage of having a higher phase constant due to the slow-wave effect. This can be translated into a potential size (and cost) reduction, since a shorter SCPW transmission line can achieve the same electrical length as a CPW and exhibit lower loss. In addition to that, the line is better shielded from the substrate reducing the risk of unwanted coupling.

A large number of transmission lines have been analyzed using Helic's RaptorX tool, for a variety of technology nodes. The metrics of characteristic impedance Zc, attenuation constant α, and phase constant β are calculated for each case and have been presented in the form of trend plots [42]. In the parametric analysis a standard 50 CPW transmission line serve as the reference design and all other variants are compared to that.

Since size reduction and hence the saving of silicon area is of paramount importance in CMOS transmission line design in order to reduce costs, a methodology [42] based on the principle of substituting λ/4 transmission line segments of initial characteristic impedance Z with equivalent lowpass structures has been developed. The aforementioned methodology has been verified through a parametric analysis of nm CMOS integrated transmission lines for mm-wave frequencies and appropriate design guidelines have been proposed. The proposed new transmission line segments, based on the proposed methodology, can be made significantly shorter by raising their characteristic impedance level to $\sqrt{2}Z$ and add shunt capacitors as loading elements.

The proposed electromagnetic parametric analysis enables the IC designer to gain understanding of both the transmission line performance and the PDK limits in terms of transmission line design. The analysis demonstrates the slow-wave effect of the SCPW line and its suitability for mm-wave designs. Finally, a semi-lumped CPW design is proposed and successfully implemented on-silicon at 80GHz, proving its superiority in terms of area reduction compared to both CPW and SCPW transmission lines. The proposed semi-lumped device is highly attractive for mm-wave nm CMOS designs due to its IC area reduction potential and moderate losses.

In the following two sub-sections a theoretical background of the transmission lines modeling along with their corresponding metrics are briefly described. Moreover a more detailed description of the geometries used in modern CMOS transmission lines is given.

9.1.1 Silicon-Integrated Transmission Lines

In electrical engineering and communications, the term transmission line is widely used for describing a propagation medium used for AC signal transmission [43].

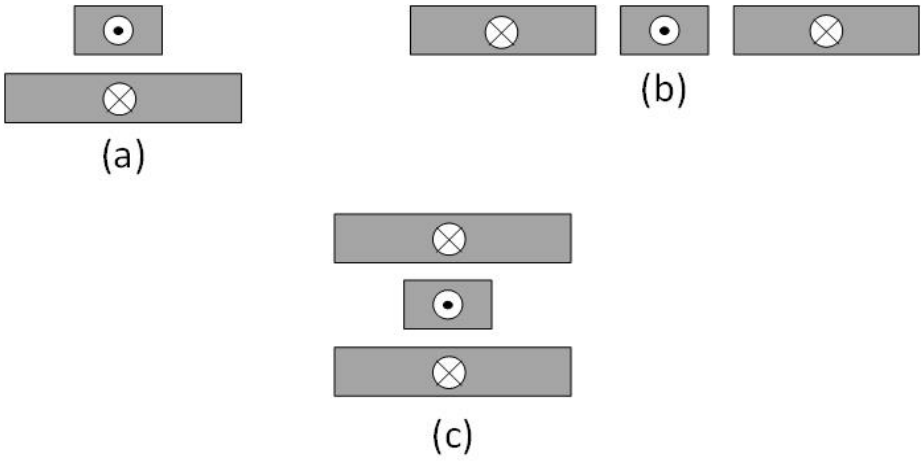


Figure 47 Cross sections of (a) Microstrip, (b) Coplanar Waveguide, (c) Stripline T-line geometries.

In its general form, a transmission line can be any type of two-wire system with signal propagation and return path. The geometries that fall under this generic description include a two-wire shielded copper cable, coaxial transmission line, thin-film printed transmission lines, waveguides, and so forth. The AC signals propagating on such transmission lines can range from the lower megahertz region up to millimeter-wave frequencies. In silicon-integrated technology, however, we face very specific limits in the geometries that can be fabricated. Designing silicon-integrated transmission lines using the given metal layers is practically limited to the geometries of **Figure 47**, which use current return paths either on the same metal layer, or at upper and lower metal layers.

In silicon-integrated transmission line design designers use either coplanar or multilayered conductors of rectangular cross sections, which are embedded in multiple dielectrics. Transmission lines, called here for simplicity T-lines, are essential building blocks in the circuit design process of silicon-integrated circuits. Their importance is derived from their versatility since T-lines can be used in multiple ways (e.g., as a high-frequency interconnect, a lumped reactive element L or C, a resonator unit, or an impedance transformer). The physics behind them are complex since T-lines are metal traces drawn over a reference plane, which in our investigation carry electromagnetic signals of a broad frequency range. The electric energy stored between the signal and ground metallization is described by a distributed capacitance C (F/m). In a similar way the magnetic energy stored on the metal trace that is subject to an alternating current translates to a varying magnetic flux, which in turn is described by a distributed inductance L (H/m).

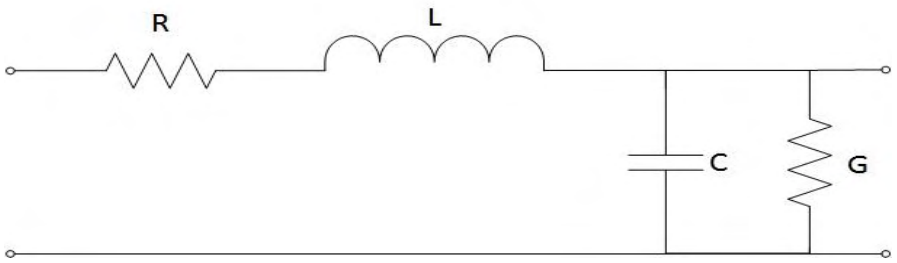


Figure 48 Transmission line model with distributed network elements.

By nature, transmission lines are conducting paths fabricated by materials of finite conductivity and are subject to ohmic conductor losses, which have a frequency-dependent behavior and can be described by a distributed resistance *R* (Ω/m). Additional losses are introduced in the signal path due to the dielectric displacement charges between the signal and reference conductor. Typically, one or multiple dielectrics will be present between them, introducing losses that are described by a distributed conductance *G* (S/m). In a schematic representation, we can describe a transmission line as a two-port network as in **Figure 48**.

Following basic network analysis [44], the governing equations that describe the AC signal propagation along the transmission line can be derived. Considering now a line voltage *V (x)* and the resulting current *I (x)* propagating along the lossy T-line in the x-direction, we can formulate the following equations.

$$\frac{\partial V(x)}{\partial x} = -(R + j\omega L)I(x)$$

$$\frac{\partial I(x)}{\partial x} = -(G + j\omega C)V(x)$$

Solving for the wave quantities *V (x)* and *I (x)* leads to the following set of equations

$$\frac{\partial^2 V(x)}{\partial x^2} - \gamma^2 V(x) = 0$$

$$\frac{\partial^2 I(x)}{\partial x^2} - \gamma^2 I(x) = 0$$

where γ is the complex propagation constant.

$$\gamma = \alpha + j\beta = \sqrt{(R + j\omega L)(G + j\omega C)}$$

The attenuation constant α is a metric of the losses on the transmission line, while the phase constant β describes the propagation delay for an AC signal passing through the transmission line. From the solution of the wave equations, we can derive an expression for the characteristic impedance *Zc* that describes the ratio of voltage to current traveling along the transmission line.

$$Z_c = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

For an ideal and lossless transmission line ($\alpha \rightarrow 0$) or a nearly lossless line with negligible *R* and *G*, we may simplify the complex propagation constant and characteristic impedance.

$$\gamma = \alpha + j\beta = j\omega\sqrt{LC}$$

$$\beta = \omega\sqrt{LC}$$

$$Z_c = \sqrt{\frac{L}{C}}$$

One may argue that the assumption of a lossless transmission line is not practical since a real-world transmission line will always exhibit loss. Nevertheless, the simplified formulae for the phase constant and characteristic impedance of a lossless line may serve us as a valuable starting point for understanding the electromagnetic performance of silicon-integrated transmission lines. Adopting this distributed element model enables us to predict the electrical transmission line metrics by observing the distributed inductance and capacitance.

9.1.2 CMOS Transmission Lines

Transmission lines can be implemented in a variety of configurations depending on the target application, the frequency range of interest, and the power-handling capabilities [43]. In the scope of our investigation we are interested in transmission line geometries suited for integration in silicon technology. Indicative layouts and cross sections of CMOS transmission lines are shown in **Figure 49**. In RF CMOS circuits transmission lines will be typically designed by using the upper layers of the back end of line (BEOL) in order to minimize the ohmic losses and allow for wider unslotted metal track widths. The most common transmission line type is the coplanar waveguide consisting of a signal metal strip and two symmetrically placed coplanar current return paths, drawn on the same metal layer. Its popularity for RF silicon designs can be explained by a number of factors. One major advantage is that the CPW geometry is compatible with onwafer measurement setups, which typically include coplanar waveguide probes with GSG configuration in two-port and GSGSG in four-port measurements. Therefore, designing CPW transmission lines for test and measurement is just a straightforward extension of the actual CPW design used in RF circuits. Another aspect that contributes to the popularity of this transmission line type is the availability of ground metallization on the same metal layer and its vicinity to 84 On-Wafer Microwave Measurements and De-Embedding the signal propagation path. This geometry makes it easy to connect other network elements like shunt loads between the signal line and the ground network [45].

The CPW transmission line is subject to electromagnetic coupling with the underlying substrate, which contributes to the total loss of this device. An evolution from the CPW transmission line is the coplanar waveguide with ground (CPWG), with an additional ground metal plane placed below the signal path. By doing so, the signal path is largely shielded from the substrate and is expected to exhibit lower losses.

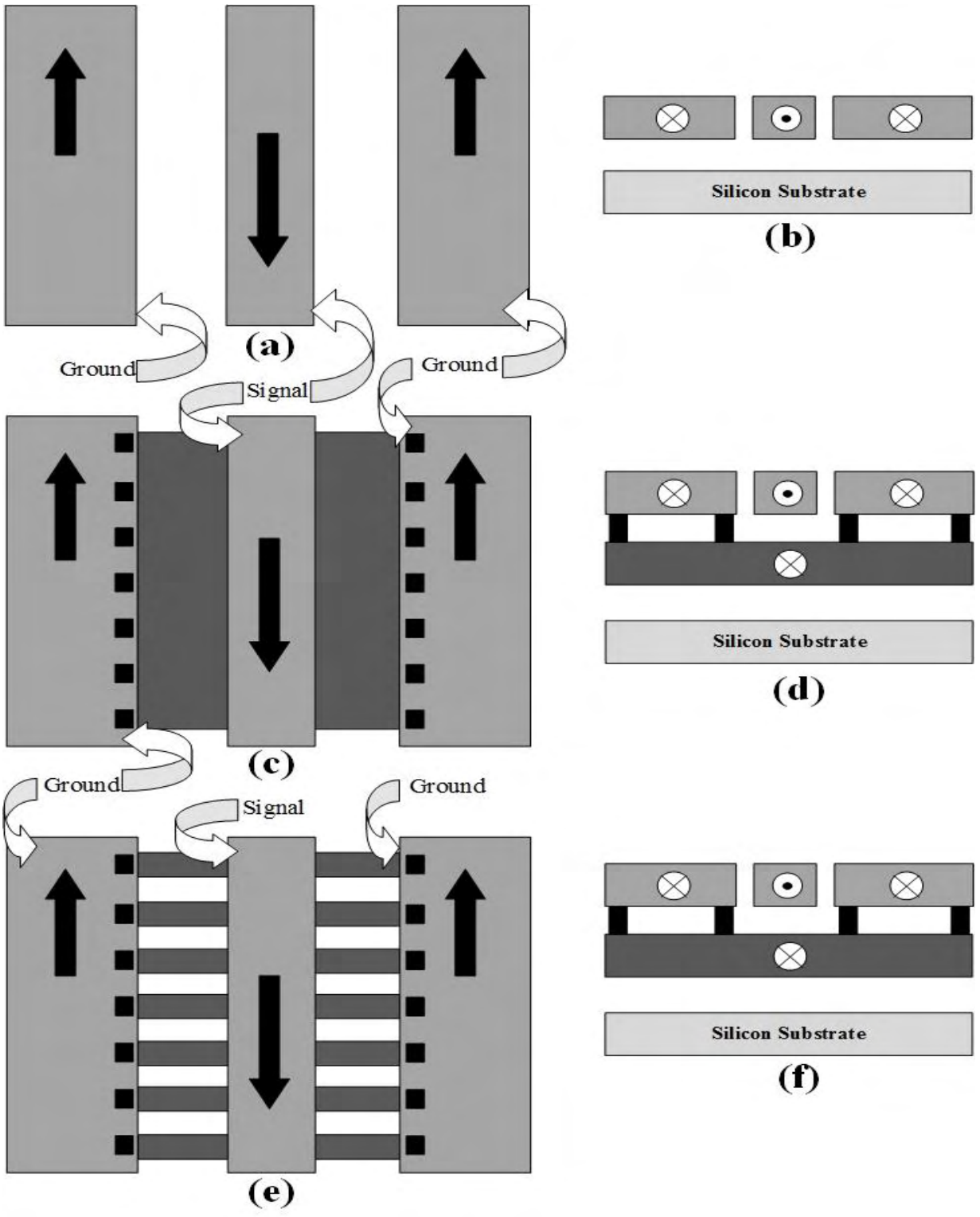


Figure 49 (a) CPW layout, (b) CPW cross section, (c) CPWG layout, (d) CPWG cross section, (e) SCPW layout, (f) SCPW cross section. Example geometries of CMOS transmission lines

However, the introduced additional ground plane contributes to both the current return path, thus the distributed inductance *L*, and to the distributed per length capacitance *C* of the transmission line. As a result, the CPWG transmission line will exhibit quite different electrical metrics than its CPW counterpart. Designing a CPWG transmission line with the proper characteristic impedance $Z_c \approx \sqrt{L/C}$ has to be done carefully.

A slightly modified version of the regular CPW is the shielded coplanar waveguide (SCPW) transmission line being a hybrid between the CPW and CPWG transmission line. Its main difference from the CPW consists of the underpasses connecting the two ground planes, which form a type of shield structure for the signal path. The shield captures more dynamic field lines and further minimizes the electromagnetic coupling with the substrate, resulting in lower loss. Another important aspect of the SCPW is the slow-wave effect that takes place. The underpasses do not essentially affect the current loop since the current continues to return preferably over the wide ground metallization on the top layer. As a result, the per-length inductance L seen for a SCPW T-line is almost identical to a CPW T-line of the same geometry. However, the metal bridges under the signal path introduce increased capacitive coupling between the signal and the ground terminals, which is translated into an increased per-length distributed capacitance C. The first effect is a drop in the characteristic impedance $Z_c \approx \sqrt{L/C}$ and an increased phase constant $\beta \approx \sqrt{LC}$ for the same physical length. This last observation is of particular interest since, for the same physical length, an SCPW transmission line will exhibit an increased phase shift compared to its CPW counterpart [46, 47].

An EM wave that travels along an SCPW appears to be slowed down compared to the CPW signal propagation. As a result thereof, the SCPW transmission line exhibits more phase shift or delay for the same length. This slow-wave effect is a promising technique for achieving physical reduction when using T-lines as building blocks of more complex networks such as filters [48], combiners/dividers [49], and phase shifters [50]. Another side effect of the SCPW is the suppression of unwanted slot-line modes that may arise when CPW T-lines have unequal wide ground planes. This is often the case for CPW T-lines with discontinuities such as bends, open stubs, and short stubs. A good practice is to use underpasses at the beginning and the end of a CPW discontinuity in order to ensure the suppression of parasitic modes propagating along the transmission line [51]. For a better understanding, let's examine the electrical performance of a coplanar waveguide transmission line with characteristic impedance of *Zc* = 50Ω, as shown in **Figure 50**. The transmission line has been designed in a typical CMOS BEOL for a characteristic impedance of *Zc* = 50Ω around 50 GHz. Some interesting points to mention here is the typical broadband behavior observed for the characteristic impedance *Zc*, the monotonic increase of the losses and the linear phase shift. The losses are dictated by the ohmic conductor losses and the electromagnetic interaction with the underlying silicon substrate. The increasing phase shift shows a linear increase until it reaches the $\lambda/2$ electrical length where it manifests a resonance behavior. It is good practice to consider the per length attenuation α (dB/mm) and per length phase constant β (deg/mm), when it comes to transmission line characterization. The reason is that using the per-length metrics allows us to compare individual transmission line types in terms of their RF performance.

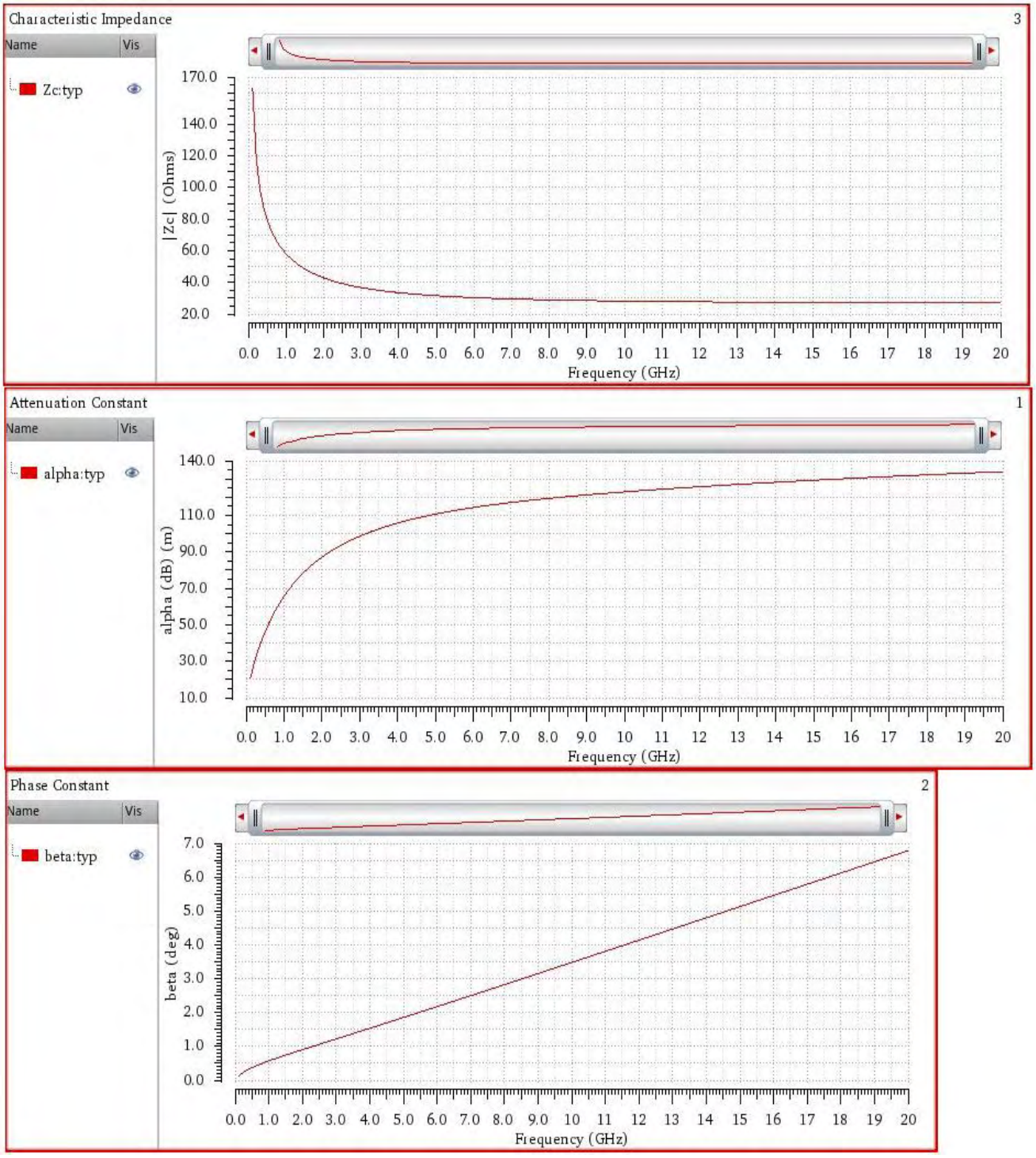


Figure 50 Transmission line responses for CPW design (Cadence Virtuoso).

9.2 Differential Transmission Line as Clock Distribution Network

To demonstrate how transmission lines perform when used as clock distribution network in modern deep submicron ICs, several CMOS transmission lines [42] are connected one after the other to build a Clock Network (**Figure 51**) [52-55]. It has been designed using structures from a Parametric Cells library (CPWG) that has been developed. Each of the library structures belongs to a Transmission Line type that has been already described. The geometry of these Parametric Cells has enhancements that make them more robust in terms of simulation results. The technology process used was in a 45nm node. The differential signals are routed in M8 along with the coplanar grounds. The bottom plane, which is also connected to ground, is routed in M6. The differential transmission lines are extremely long (750.454 um), lying over the digital part of the design, covering the entire chip in the horizontal direction. The clock period is 0.4ns.

Figure 51 Clock Network using Differential Transmission Line (Top View).

To help avoiding magnetic coupling between the parallel signals, the proposed geometry switches the signals with each other, as depicted in the 3D view of the clock transmission line, in **Figure 52**.

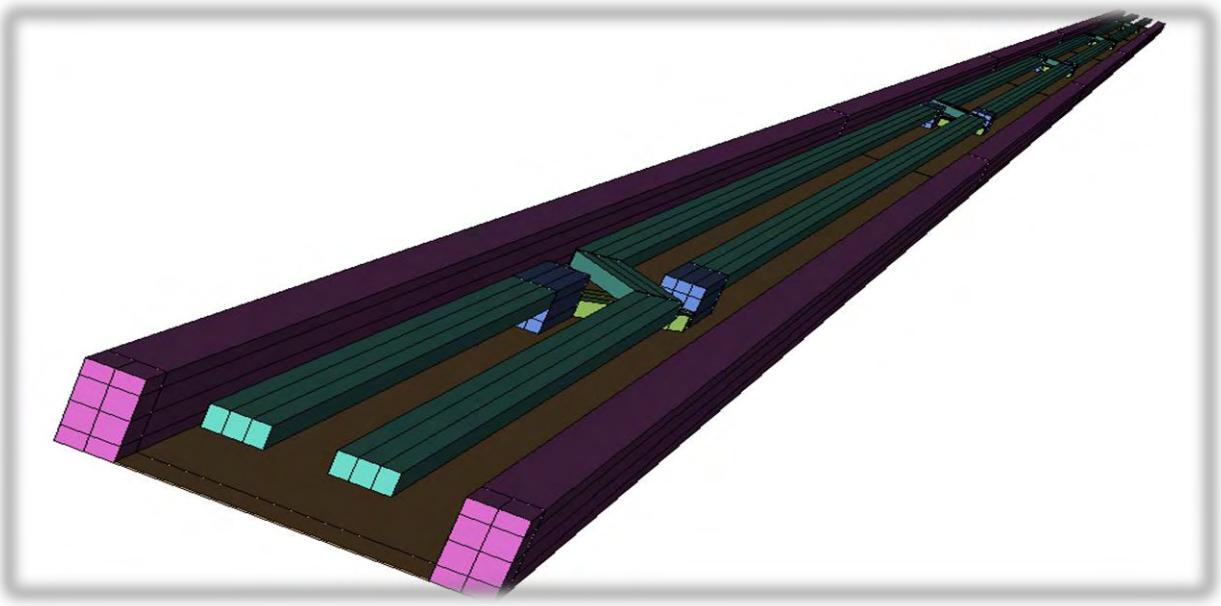


Figure 52 Clock Network using Differential Transmission Line (3D View).

The coplanar grounds are connected with stacked metal stripes from M8 down to M6 creating a closed shield, which encapsulates only the differential lines. To study the behavior of this modern type clock network, the geometry in **Figure 52** has been modeled using HELIC's RaptorX™, configuring the extraction, to build a detailed model. The extracted netlist is then imported through a schematic symbol, in a schematic testbench, in Cadence Virtuoso. In the model (schematic symbol), input ports are considered the inputs of the differential lines (left edge of the lines) and output ports are considered the outputs of the differential lines (right edge of the lines). Four additional ports are used to connect the shield-paths to ground. In the testbench an ideal voltage source (pulse: 0.5V) is connected between the input ports of the model, generating this way a differential signal (static wave) across the transmission lines. The output ports are connected to 50Ω resistor. The following waveforms in **Figure 53** and **Figure 54** (zoom-in) depict the simulation results. **Figure 53 (a)** depicts the differential waveform of the two signals (signal 1 – signal2) and **Figure 53 (b)** the waveforms of each signal separately. In **Figure 53 (a)** the cyan is the ideal differential waveform, produced by the ideal voltage source, on the inputs of the model and the blue is the corresponding differential waveform on the outputs of the model. In **Figure 53 (b)**, the positive waveforms correspond to the first input and its corresponding output, while the negative waveforms correspond to the second input and its corresponding output.

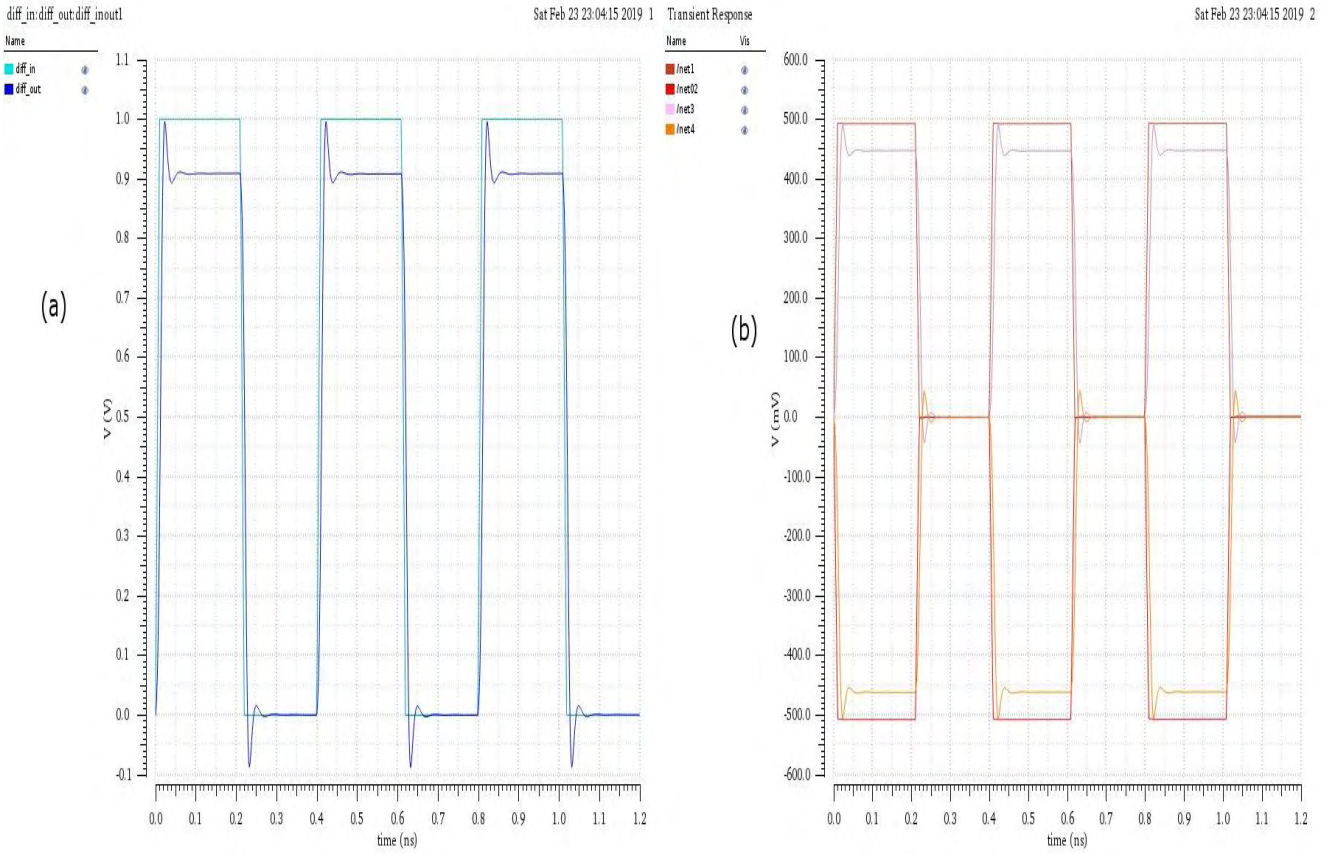


Figure 53 Differential Input signal vs Differential Output signal.

The following **Figure 54** zooms in the first period of the clock signal in **Figure 53 (a)**. A horizontal axis is used in order to check the delay of the pulse from the inputs to the outputs. As we explained the delay is proportional to the length of the transmission line. In this case for the 80% of the pulse the delay is 8.13534 ps on the positive edge of the pulse and 3.8204 ps on the negative edge of the pulse.

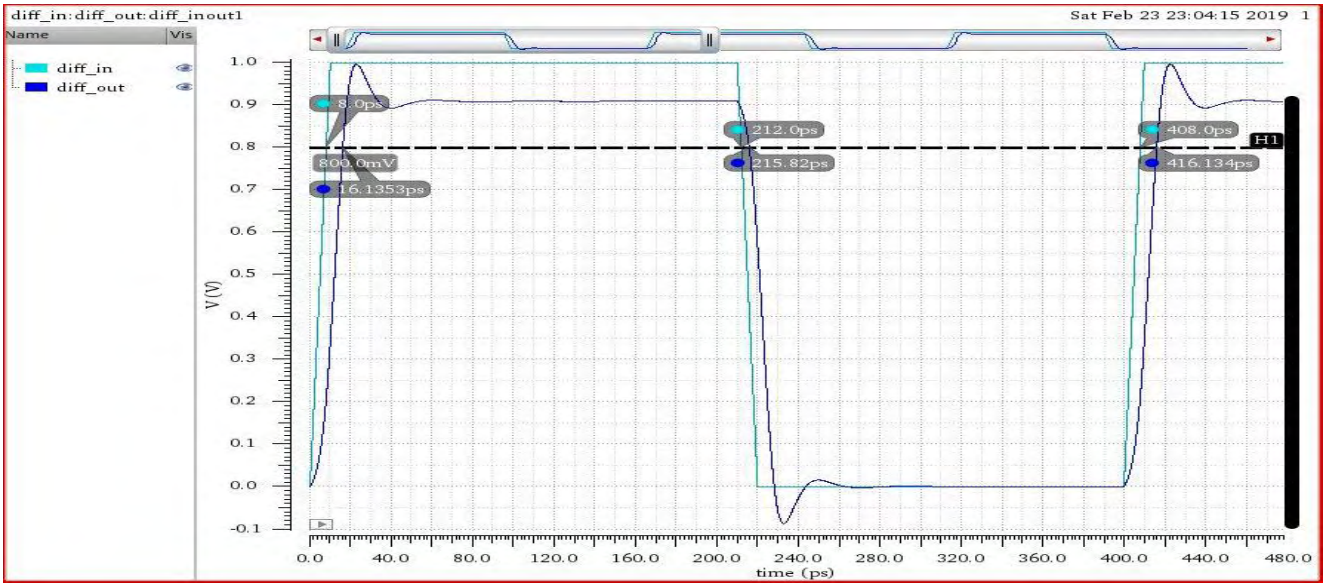


Figure 54 Delay and Pulse Width Attenuation on Differential Output signal (80% pulse width).

Another obvious effect is a ~10% attenuation of the pulse width, which is an expected result, due to the resistance of the transmission line. In **Figure 55** is depicted the same waveform aligning the ruler at 1.0 volt in order to view the delay at the 100 % of the pulse width.

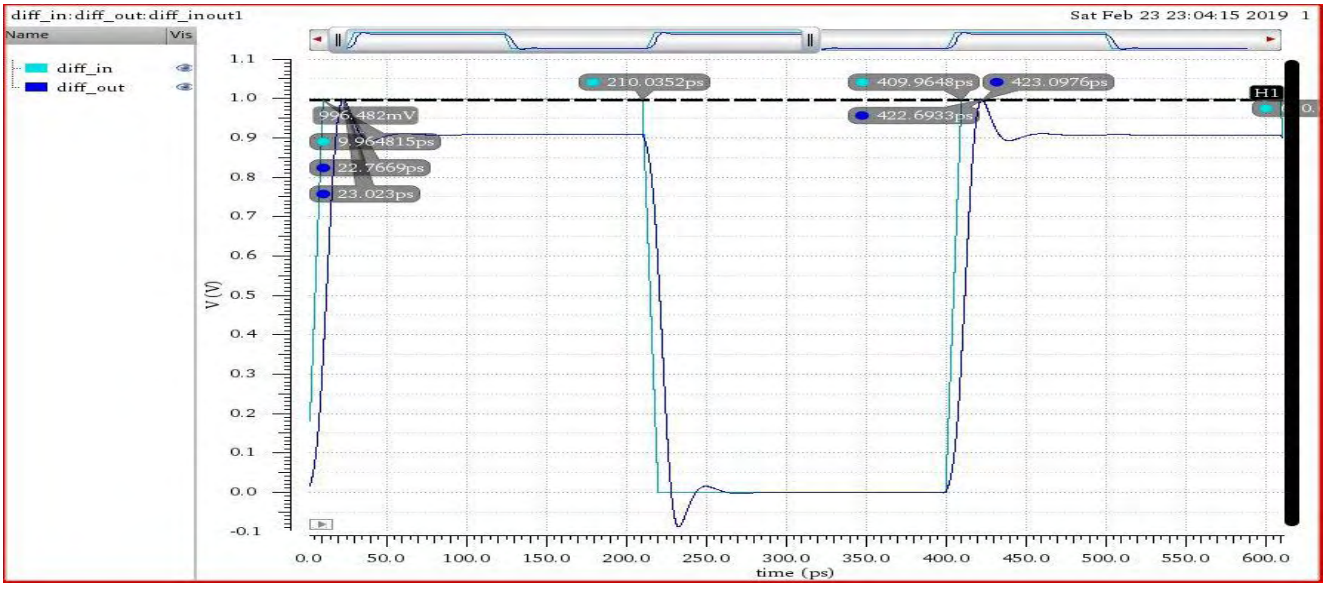


Figure 55 Delay and Pulse Width Attenuation on Differential Output signal (100% pulse width).

10 Conclusions

To sum up all the work that has been done during this PhD thesis, two version of a highly accurate Power Integrity (PI) methodology for the estimation of the worst-case voltage-drop have been developed. The proposed methodology achieves accuracy within 3% of SPICE simulators, orders of magnitude faster having the capacity to consume industrial size designs, offering this way the opportunity for statistical analysis of these results. The statistical analysis is performed by a fast and fully parallelizable statistical prediction engine, developed in C++ code, based on a well-established theory called Extreme Value Theory (EVT). Extreme Value Theory is used for the estimation of the unknown maximum of a related population from its samples given a confidence interval. Either applied on the currents drawn by the IC's devices (version 1) or directly on the voltages reported on the Power Supply Network (version 2) the statistical prediction engine accurately predicts the worst-case voltage waveforms (both voltage-drops for the Power Supply Network and ground-bounce for the Ground Supply Network). In a design era where the nominal voltage levels have been significantly decreased forcing the margin between logic '0' and logic '1' to be very small and the size of modern ICs in terms of area and devices has extremely increased, increasing this way the power consumption, make voltage-drop a dominant effect and thus, power integrity analysis more indispensable than ever before.

Experiments verifying the direct and significant impact of the voltage drop effect on timing, both the delay and the critical path structure have been set. Two methodologies have been implemented both for Static Timing Analysis (STA) and Dynamic Timing Analysis (DTA). Both timing analysis methodologies make use of the results of the aforementioned power integrity methodology bring extra accuracy on both of them. The proposed static timing methodology relaxes in a very safe way the voltage level annotated on devices during timing analysis showing significant extra positive margin in the results. Moreover a totally novel methodology for Statistical Dynamic Timing Analysis (SDTA) has been developed away from all existing proposals in the field that search for the exact one input vector pair that produces the worst-case delay taking into account the voltage drop effect.

Since power and timing are highly correlated, as showed in this thesis, the question whether there is a method to estimate the probability, the performance of an IC to exceed a specification limits in either Power or Delay (or both) becomes of high importance. An also novel approach for process variation has been proposed. This approach performs statistical analysis jointly for both delay and power with respect to process variation answering the aforementioned question.

At last new design techniques using IC Transmission lines in modern designs have been studied along with the best practices and the appropriate transmission line types for usage in designing Clock distribution networks. The results are very import and expose a delay on the clock signal that should accounted in the timing analysis for the best accuracy.

To conclude, a holistic approach is proposed, coupling the two effects of high importance that take place during modern deep sub-micron ICs operation and affect their performance that until know were investigated as decoupled. The results are very encouraging. A process variation methodology has also been studied with a totally novel approach answering an important question, completing this way the proposed holistic solution for Timing and Power Integrity Analysis.

11 Appendices

11.1 Appendix A

11.1.1 NLDM Model

There are three main ways to precharacterize the delay of all the unit elements assembling a design. The simplest and the same time the oldest one, is the **Non-Linear-Delay-Model** (**NLDM**) which makes use of a table of delay values. Each delay value on this one, two or three dimensional tables is a function of the input transition time and the total capacitance of a specific output port which changes logic value, either from logic zero to logic one or from logic one to logic zero. The unit elements of a design usually are logic gates and memory block but there may be any type of self-sustained digital or mixed signal block having digital input and output signals, like a large sensor or a decoder, which may be used as a part, serving specific logic functionality in a larger design, connected with logical signals. In the Liberty format, the NLDM model is described using Table templates. Table templates store common table information that multiple lookup tables can use. A table template specifies the table parameters and the breakpoints for each axis. At each template a name has been assigned, so that lookup tables can refer to it. A general description of a table template is the following one:

lu_table_template(name) {
variable_1 : value;
variable_2 : value;
variable_3 : value;

index_1 ("float, ..., float");
index_2 ("float, ..., float");
index_3 ("float, ..., float");
}

The table template specifying timing delays can have up to three variables (*variable_1*, *variable_2*, and *variable_3*). The variables indicate the parameters used to index into the lookup table along the first, second, and third table axes. The parameters are the input transition time of a constrained pin, the output net length and capacitance, and the output loading of a related pin. The following is a list of the valid values (divided into sets) that you can assign to a table:

• Set 1:
input_net_transition

• Set 2:
total_output_net_capacitance
output_net_length
output_net_wire_cap
output_net_pin_cap

• Set 3:
- related_out_total_output_net_capacitance
- related_out_output_net_length
- related_out_output_net_wire_cap
- related_out_output_net_pin_*cap*

The values you can assign to the variables of a table specifying timing delay depend on whether the table is one-, two-, or three-dimensional. For every table, the value you assign to a variable must be from a set different from the set from which you assign a value to the other variables. For example, if you want a two-dimensional table and you assign *variable_1* with the *input_net_transition* value from set 1, then you must assign *variable_2* with one of the values from set 2.

Breakpoints:

The index statements define the breakpoints for an axis. The breakpoints defined by *index_1* correspond to the parameter values indicated by *variable_1*. The breakpoints defined by *index_2* correspond to the parameter values indicated by *variable_2*. The breakpoints defined by *index_3* correspond to the parameter values indicated by *variable_3*. The index values are lists of floating-point numbers greater than or equal to 0.0. The values in the list must be in increasing order. The size of each dimension is determined by the number of floating-point numbers in the indexes. You must define at least one *index_1* in the *lu_table_template* group. For a one-dimensional table, use only *variable_1*

The rules for specifying lookup tables, apply to delay arcs as well as to constraints. This is the syntax for lookup table groups:

```
lu_table(lu_table_template) {
        index_1 ("float, ..., float");
        index_2 ("float, ..., float");
        index_3 ("float, ..., float");
        values("float, ..., float", ..., "float, ..., float");
}
```

The following rules apply to lookup table groups:

1. Each lookup table has an associated name for the *lu_table_template* it uses. The name of the template must be identical to the name defined in a library *lu_table_template* group.
2. You can overwrite any or all of the indexes in a lookup table template, but the overwrite must occur before the actual definition of values.
3. The delay value of the table is stored in a values attribute.
   3.1. Transition table delay values must be *0.0* or greater. Propagation tables and cell tables can contain negative delay values.
   3.2. In a one-dimensional table, represent the delay value as a list of *nindex_1* floating-point numbers.
   3.3. In a two-dimensional table, represent the delay value as *nindex_1 * nindex_2* floating-point numbers.
   3.4. If a table contains only one value, you can use the predefined scalar table template as the template for that timing arc.
4. Each group of floating-point values enclosed in quotation marks represents a row in the table.
   4.1. In a one-dimensional table, the number of floating-point values in the group must equal *nindex_1*.
   4.2. In a two-dimensional table, the number of floating-point values in a group must equal *nindex_2* and the number of groups must equal *nindex_1*.
   4.3. In a three-dimensional table, the total number of groups is *nindex_1 * nindex_2* and each group contains as many floating-point numbers as *nindex_3*. In a three-dimensional table, the first group represents the value indexed by the (*1, 1, 1*) to the (*1, 1, nindex_3*) points in the index. The first *nindex_2* groups represent the value indexed by the (*1, 1, 1*) to the (*1, nindex_2, nindex_3*) points in the index. The rest of the groups are grouped in the same order.

Each reference lu_table_templates has a specific name and describes the number of dimensions of the specific table, along with the size of each dimension, the type of variable that corresponds to each dimension. The reference lu_table_templates are located at the top of the liberty file. During all the precharacterizations of all PDKs used for the experiments of this thesis, two dimensional tables with *variable_1* representing *input_net_transition* and *variable_2* representing *total_output_net_capacitance* were used for the NLDM model. The size of these tables for the precharacterizations performed, varies from 7x7 (49 values) to 9x9 (81 values). A simple example of a reference lu_table_template used for NLDM representation in a Nangate-45nm PDK is the following:

```
lu_table_template(tmg_ntin_oload_7x7) {
    variable_1 : input_net_transition ;
    variable_2 : total_output_net_capacitance ;
    index_1("1, 2, 3, 4, 5, 6, 7");
    index_2("1, 2, 3, 4, 5, 6, 7");

}
```

Let's now describe the transitions delay and how it is stored in Liberty NLDM format. The transition delay of an element is the time it takes the driving pin to change state. Transition delay attributes represent the resistance encountered in making logic transitions. The components of the total delay calculation depend on the timing delay model used. Include the transition delay attributes that apply to the delay model you are using. Transition time is the time it takes for an output signal to make a transition between the high and low logic states. It is computed by table lookup and interpolation. Transition delay is a function of capacitance at the output pin and input transition time.
These *timing* group attributes provide valid lookup tables for delay arcs:
• *cell_rise*
• *cell_fall*
• *rise_propagation*
• *fall_propagation*
• *retaining_rise*
• *retaining_fall*
• *retain_rise_slew*
• *retain_fall_slew*

Note:
For timing groups with timing type clear, only fall groups are valid. For timing groups with timing type preset, only rise groups are valid.

There are two methods for defining delay arcs.
Method 1
To specify cell delay independently of transition delay, using one of these timing groups attributes as lookup table:
• *cell_rise*
• *cell_fall*

Method 2
To specify transition delay as a term in the total cell delay, using one of these timing groups attributes as lookup table:
• *rise_propagation*
• *fall_propagation*

In NLDM precharacterizations performed in the context of this thesis the first method was used. A simple example of a lu_table, with lu_table_template, corresponding to the previous example, used for NLDM representation of the output pin delay with related input pin *"A1" for* the AND2_X1 gate, in Nangate-45nm PDK is the following:

```
cell_rise(tmg_ntin_oload_7x7) {
        index_1("0.0023476 , 0.009448, 0.034372 , 0.081968 , 0.15612 , 0.26016 , 0.39708");
        index_2("0.00036562, 0.001893, 0.0037861, 0.0075722, 0.015144, 0.030289, 0.060577");
        values("0.053925, 0.0666195, 0.080675, 0.1075221, 0.1598424, 0.2636309, 0.4701039",\
        "0.0564212, 0.0691052, 0.0831807, 0.1099709, 0.1623386, 0.2659621, 0.4726477",\
        "0.0654954, 0.078161, 0.0922006, 0.1189463, 0.1713543, 0.2751478, 0.4819662",\
        "0.0828432, 0.0954383, 0.109412, 0.1360814, 0.1884305, 0.2921739, 0.4992483",\
        "0.1045581, 0.1174121, 0.131499, 0.1582021, 0.2105062, 0.3142604, 0.5213444",\
        "0.1275287, 0.1407105, 0.1548618, 0.1815967, 0.23394, 0.3378764, 0.544901",\
        "0.1508141, 0.1647089, 0.1790113, 0.2056832, 0.2578887, 0.361778, 0.5687382");
}
cell_fall(tmg_ntin_oload_7x7) {
        index_1("0.0023476 , 0.009448, 0.034372 , 0.081968 , 0.15612 , 0.26016 , 0.39708");
        index_2("0.00036562, 0.001893, 0.0037861, 0.0075722, 0.015144, 0.030289, 0.060577");
        values("0.0671911, 0.0747577, 0.0818757, 0.0932844, 0.1123513, 0.1462065, 0.2105781",\
        "0.0698464, 0.0775033, 0.0846284, 0.0960282, 0.1150992, 0.1489594, 0.2133694",\
        "0.0794155, 0.087051, 0.0941409, 0.1055707, 0.1246459, 0.1585183, 0.2229041",\
        "0.0991071, 0.106718, 0.1138129, 0.1252673, 0.144349, 0.1782423, 0.2426224",\
        "0.1297691, 0.1375174, 0.1446957, 0.1562216, 0.1753812, 0.2093072, 0.273734",\
        "0.1662461, 0.1748672, 0.1826304, 0.1948435, 0.2145771, 0.2488012, 0.3132811",\
        "0.2051708, 0.2148414, 0.2235086, 0.2365854, 0.2572754, 0.2923314, 0.35719");
}
```

Where *index_1* is the input transitions in *ns*, *index_2* is the total output capacitance of the output pin of the gate and *values* the delay of the output pin. In the case where the input transition and the output capacitance do not have an exact match in the lu_table (which is the most common case) then an interpolation between the closest values, exist in the table, is performed to estimate the delay at each gate. An example of the delay calculation using NLDM method follows.

Consider an Inverter driving gates with a total capacitance of 0.1pf. The Inverter has been precharacterized through NLDM and the corresponding cell_fall, cell_rise, transition_fall and transition_rise matrices are the following. We use both for falling and rising input transition 0.01ns delay. To find out the exact value in each matrix we use bilinear interpolation as described in Apendix I. For example matrix cell_fall does not include the exact combination for our example which is Output Capacitance 0.1pf and Input Transition 0.01ns. To find out the fall delay for this combination we have to interpolate between the highlighted (light grey background) values of the matrix. To find the value of the unknown $\mathbf{z_4}$ in matrix cell_fall for our example, one may consider the points of Bilinear Interpolation example to be the following:

$A(x_0=0.0075722, y_1=0.156120, z_0=0.1753812)$ $C(x_1=0.015144, y_1=0.156120, z_1=0.2145771)$

$E(x_2=0.01, y_2=0.1, \mathbf{z_4}=\mathbf{?})$

$B(x_0=0.0075722, y_0=0.081968, z_2=0.1562216)$ $D(x_1=0.0.15144, y_0=0.081968, z_3=0.1948435)$

cell_fall

Output Cap/ Input Trans	**0.034372**	**0.081968**	**0.15612**	**0.26016**
0.0037861	*0.0941409*	*0.1055707*	*0.1246459*	*0.1585183*
0.0075722	*0.1138129*	*0.1562216*	*0.1753813*	*0.2093072*
0.0151440	*0.1826304*	*0.1948435*	*0.2145771*	*0.2488012*
0.0302890	*0.2235086*	*0.2365854*	*0.2572754*	*0.2923314*

0.173309

cell_rise

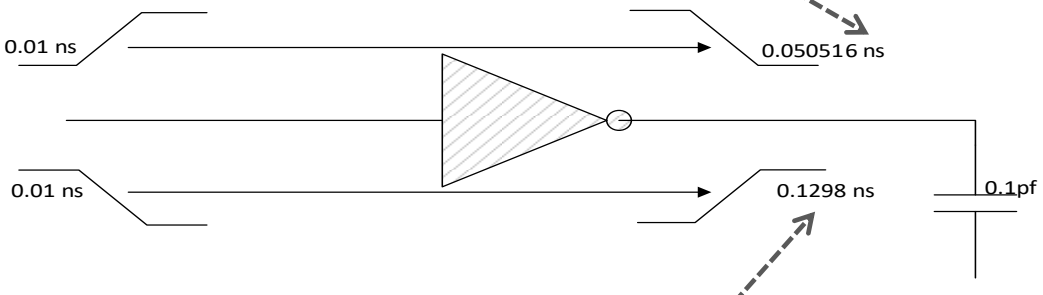
Output Cap/ Input Trans	**0.034372**	**0.081968**	**0.15612**	**0.26016**
0.0037861	*0.0922006*	*0.1189463*	*0.1713543*	*0.2751478*
0.0075722	*0.131499*	*0.1582021*	*0.2105062*	*0.3142604*
0.0151440	*0.1548618*	*0.1815967*	*0.233940*	*0.3378764*
0.0302890	*0.1790113*	*0.2056832*	*0.2578887*	*0.361778*

0.178425

fall_transition

Output Cap/ Input Trans	**0.034372**	**0.081968**	**0.15612**	**0.26016**
0.0037861	*0.0316452*	*0.044392*	*0.0692512*	*0.1197682*
0.0075722	*0.0315970*	*0.0443572*	*0.0691262*	*0.119700*
0.015144	*0.0322446*	*0.0448106*	*0.0694564*	*0.1198966*
0.030289	*0.0362216*	*0.047949*	*0.071500*	*0.1207858*

0.050516 ns



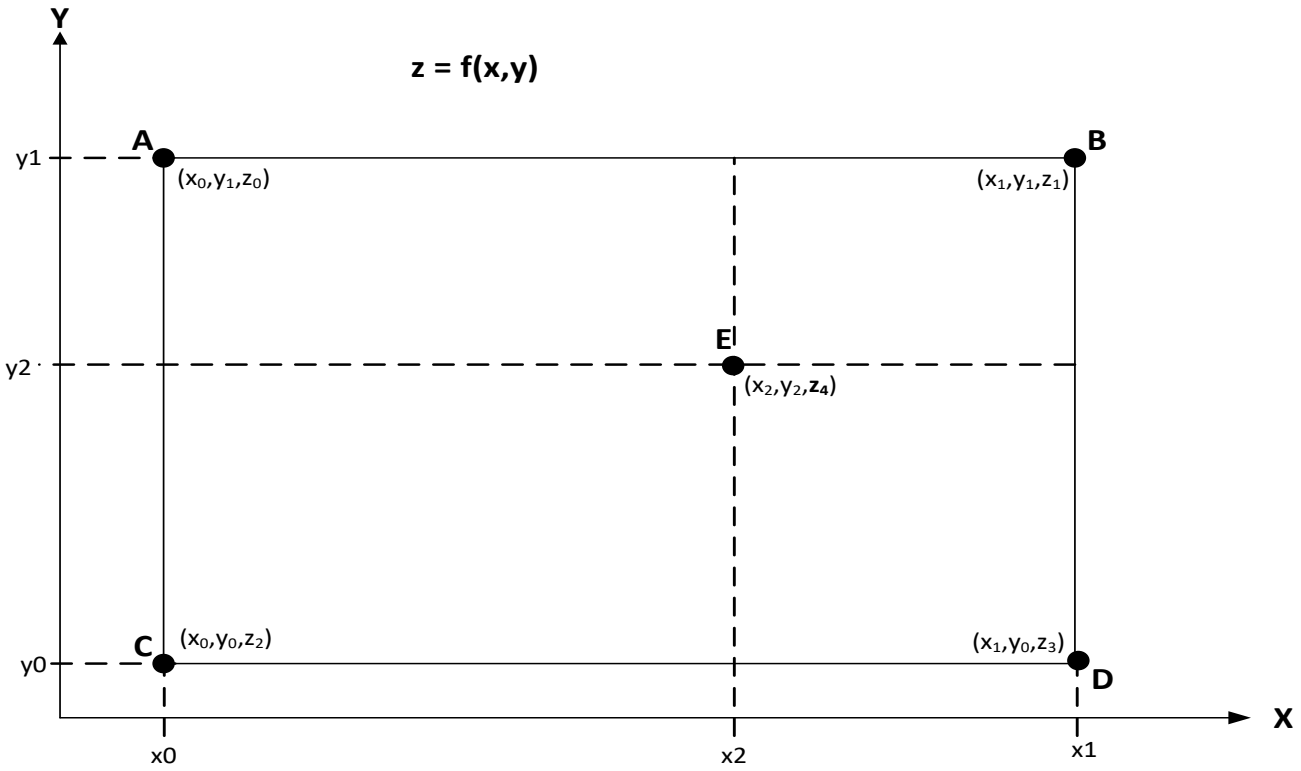
rise_transition

Output Cap/ Input Trans	**0.034372**	**0.081968**	**0.15612**	**0.26016**
0.0037861	*0.0627592*	*0.1075476*	*0.1982504*	*0.3806314*
0.0075722	*0.0628974*	*0.1076328*	*0.1985412*	*0.3808032*
0.015144	*0.0636004*	*0.1078734*	*0.1985682*	*0.3809548*
0.030289	*0.0646856*	*0.1087900*	*0.1987328*	*0.3811624*

Fall delay =0.173309 ns / Rise delay = 0.178425 ns

Fall transition = 0.050516 ns / Rise transition = 0.1298 ns

11.1.2 Bilinear Interpolation



The four normalized areas, ***Na***, ***Nb***, ***Nc***, and ***Nd***, each diagonally opposite from its naming rectangle vertex, are given by:

$$N_a = \frac{(x_1 - x_2) \cdot (y_2 - y_0)}{(x_1 - x_0) \cdot (y_1 - y_0)}$$
$$N_b = \frac{(x_2 - x_0) \cdot (y_2 - y_0)}{(x_1 - x_0) \cdot (y_1 - y_0)}$$
$$N_c = \frac{(x_1 - x_2) \cdot (y_1 - y_2)}{(x_1 - x_0) \cdot (y_1 - y_0)}$$
$$N_d = \frac{(x_2 - x_0) \cdot (y_1 - y_2)}{(x_1 - x_0) \cdot (y_1 - y_0)}$$

Then $\mathbf{z_4}$ is computed by the equation: $z_4 = \mathrm{z_0} \cdot \mathrm{N_a} + \mathrm{z_1} \cdot \mathrm{N_b} + \mathrm{z_2} \cdot \mathrm{N_c} + \mathrm{z_3} \cdot \mathrm{N_d}$

11.2 Appendix B

11.2.1 Composite Current Source Timing Model

11.2.1.1 Introduction

Accurate delay calculation is critical for timing closure of complex digital designs. At 90nm and below, physical effects and design styles present new challenges for delay calculation. Top-level interconnect is becoming more resistive with narrower metal widths, resulting in cases where the interconnect impedance is much greater than the drive resistance of the driving cell. Analysis is needed across a wide range of Vdd values to support dynamic IR drop effects, and low-power design styles including voltage islands and dynamic voltage/frequency scaling. Inverted temperature dependence at low voltages requires analysis at intermediate temperature values.

A delay model is needed that enables accuracy close to circuit simulation, but with fast calculation to support flat analysis of the largest designs. The model must support calculation of cell delay, interconnect delay, pin slew (also called "transition time") and input pin capacitance for stages including detailed parasitics. Synopsys' Composite Current Source model is described for timing (CCS Timing) which addresses these needs and future delay calculation needs [56-59].

Delay calculation is performed for one stage at a time, where a stage consists of the driving cell arc, the output RC network and the capacitance of the network load pins. The goal is to compute the response at the driver output and at the network load pins, given an input slew or waveform at the driver input, as shown in Figure 1. The computed responses are then used to determine the cell delay of the driver and the input slews at the load pins.

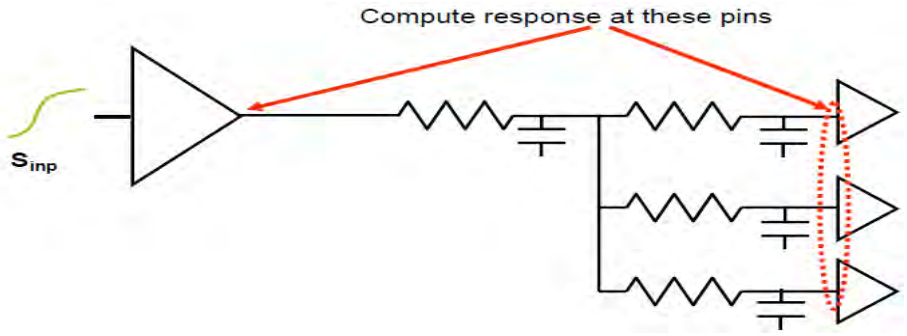


Figure 56: Stage Delay Calculation

To perform stage delay calculation efficiently, three models are created:

- The driving cell arc is replaced by a driver model
- The interconnect RC network is replaced by a reduced order model (such as Block Arnoldi)
- The load pins are replaced by a receiver model

These models are depicted in Figure 2.

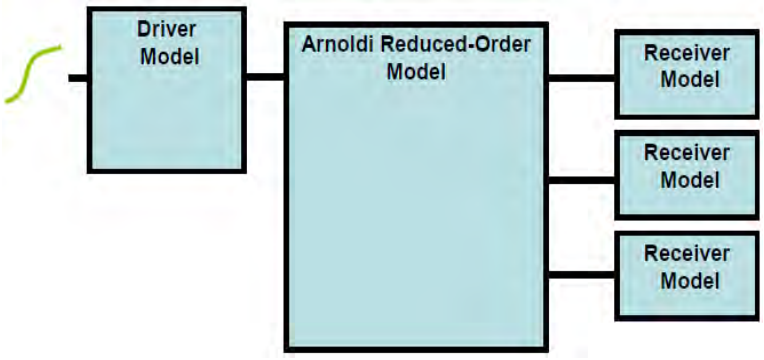


Figure 57: Stage represented by driver model, ROM and receiver models.

Note that the receiver model must represent the complex input capacitance of a cell input pin. The transistors do not present a constant input capacitance to a driver. The equivalent capacitive load (from I = C * dv/dt) can vary depending on the rise/fall direction of the transition, the input slew at the pin, the output load, and the state of the cell. In addition, this capacitance can change during the transition. The receiver model must be able to represent all these effects.

11.2.1.2 Previous Approaches

Thevenin and Norton Models

Previous driver models used either a time-dependent voltage source in a series with a resistor (Thevenin model) or a time-dependent current source in parallel with a resistor (Norton model). The resistor in those models is typically referred to as the "drive resistor" and is used to express the timing arc's sensitivity to output capacitance, whereas the waveform shape itself is primarily expressed by the voltage or current source.

Refinements to these models to account for complex aspects of transistor behavior have typically dealt with making the time-dependent nature of the voltage/current source more complex. Other approaches have dealt with multiple drive resistances and arbitrary dynamic impedances.

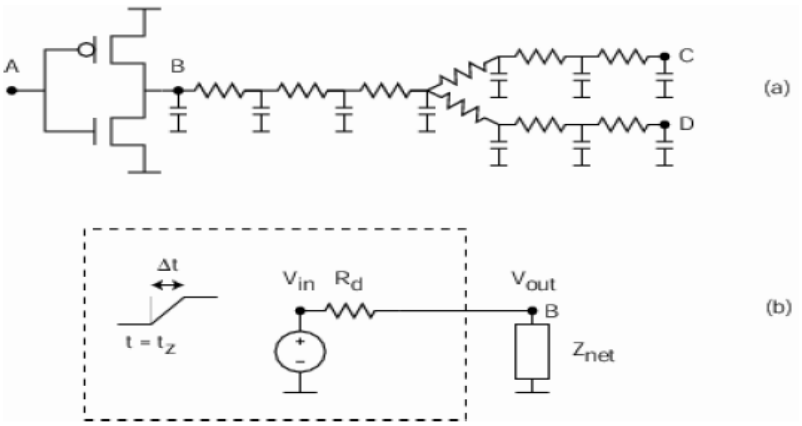


Figure 58: The Rd << Znet problem. (a) shows a transistor circuit driving a detailed parasitic network at node 'B'. (b) The network presents an impedance Znet to the Thevenin driver model. When Rd << Znet, Vout approaches Vin and the driver model can lose accuracy.

Unfortunately, a major limitation occurs when conventional models are used to drive an interconnect network with an impedance Z_{net} much greater than the drive resistance R_d. Consider the Thevenin model driving a detailed parasitic network as shown in Figure 3. Note that the circuit forms a voltage divider with

$$\frac{Vout}{Vin} = \frac{Znet}{Rd+Znet}$$

which approaches unity when $R_d << Z_{net}$. This points out that a driver model based upon a drive resistance (or arbitrary impedance) that is set independent of the network load will be ineffective in this regime. Since the transistor behavior deviates from the Thevenin voltage source nearest the power rails, this situation is usually worst when the network delay is greater than the output transition time.

Previous Receiver Models

The traditional receiver model is a single value of capacitance for an input pin. More recently, separate values have been allowed for rising vs. falling transitions, and a min/max range has been introduced that can bind the complex capacitance effects, but which leads to pessimism during analysis. Using a single capacitance value for the entire transition results in inaccuracy for single-stage cells where the Miller effect is significant, affecting the calculation of both cell delay and slew. Figure 4 shows that the voltage waveform for the input of a single-stage cell, such as an inverter, cannot be approximated well by any single capacitance value.

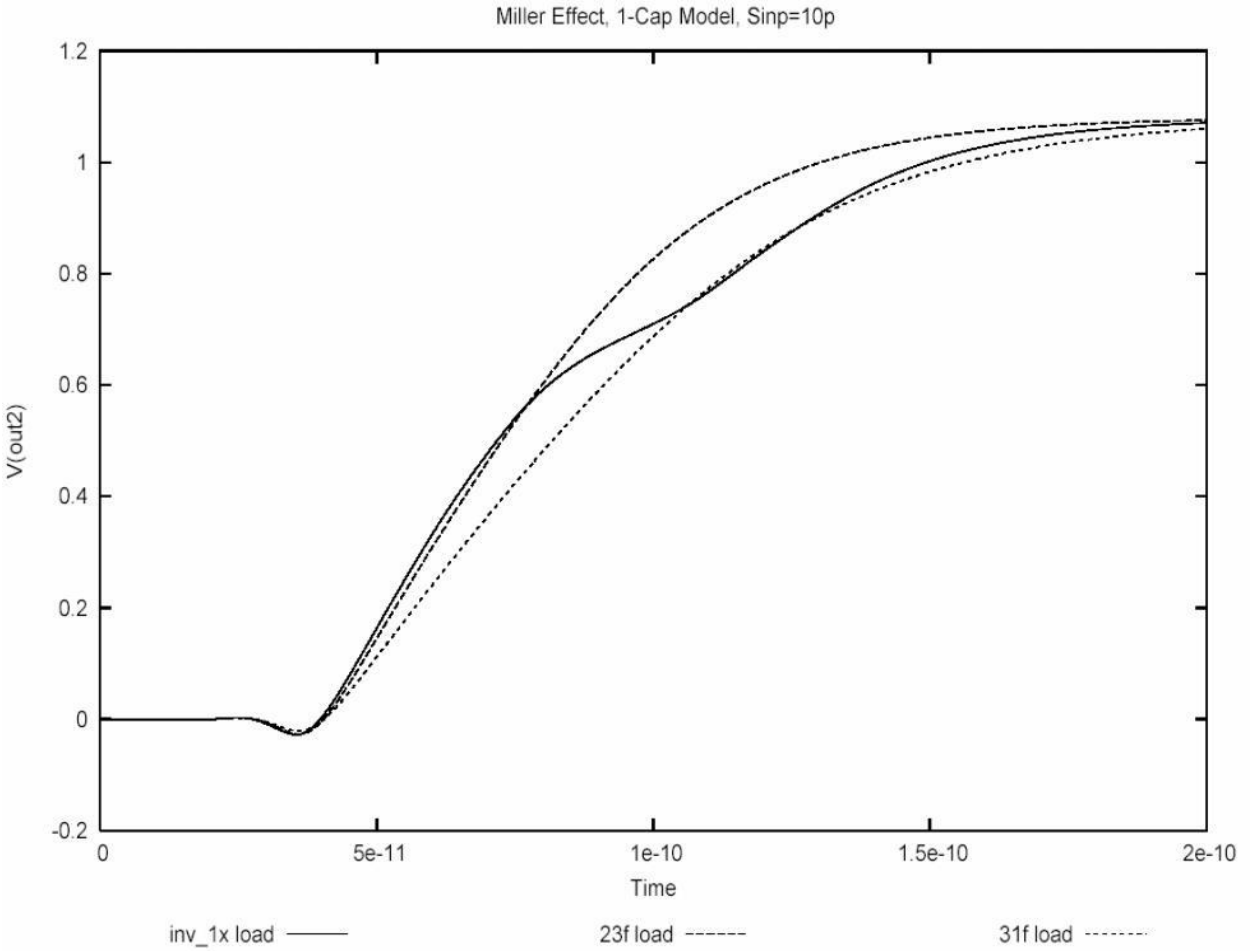


Figure 59: A single capacitance value is insufficient when Miller effect is large.

11.2.1.3 CCS Timing Model

CCS Timing consists of a driver model and a receiver model. The driver model describes how a timing arc propagates a transition from input to output, and how it can drive arbitrary RC networks. The receiver model describes the capacitance that an input pin presents to driving cells.

The CCS Timing driver model is a time and voltage dependent current source with an essentially infinite drive-resistance, which provides high accuracy even when R_d is much less than Z_{net}. The model achieves this accuracy by mapping the arbitrary transistor behavior for lumped loads to the behavior for an arbitrary detailed parasitic network, instead of modeling the transistor behavior. The following figure illustrates how the mapping algorithm basically works.

Consider a set of pre-characterization measurements of the output current as a function of time for a specific input slew and a set of output capacitances (Figure 5). When these currents are applied to their respective capacitances, the voltage waveforms can be reconstructed. If an output capacitance is presented, which was not pre-characterized with, then interpolation can be performed between the currents to predict the resulting waveform. Similarly, if an input slew is presented, that was not used for pre-characterizing, and interpolation can also take place.

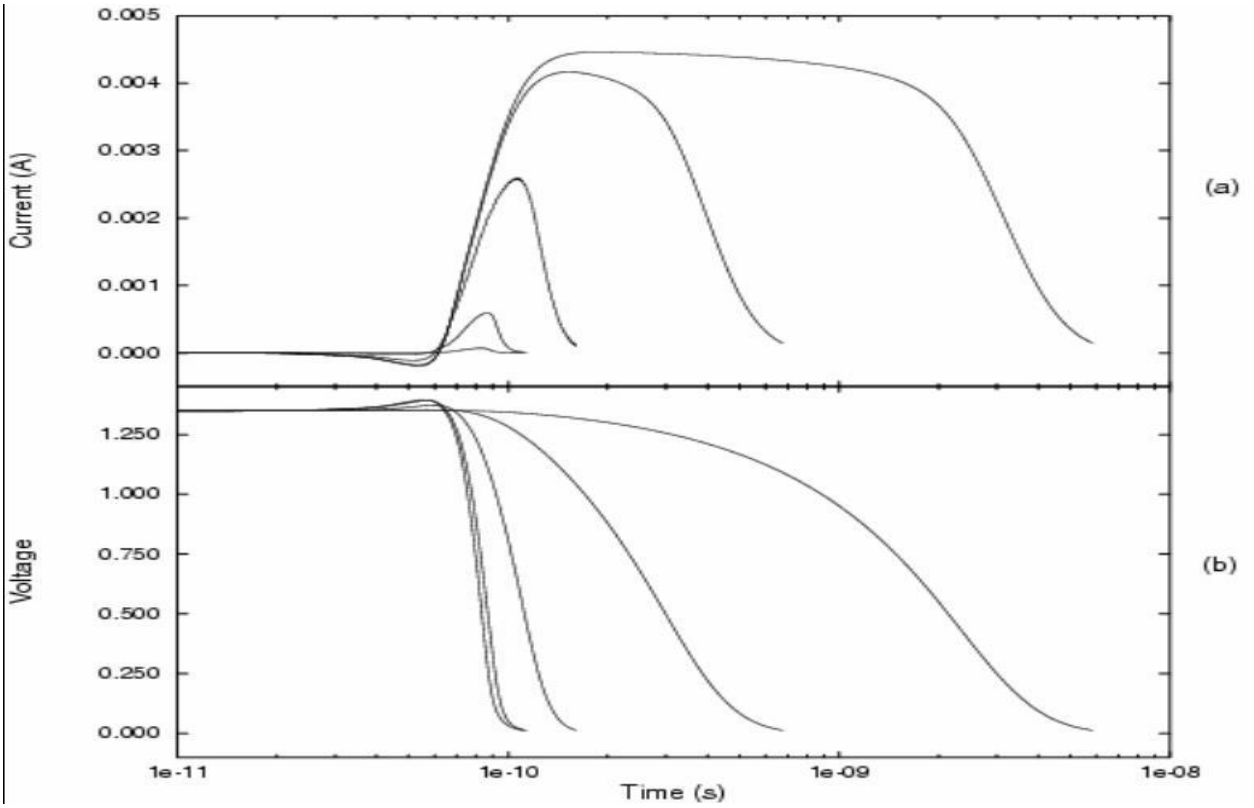


Figure 60: Output current and voltage responses for a timing arc. Transition-level simulation results are shown for different values of load capacitance (1fF, 10fF, 100fF, 1pF, 10pF). (a) Inverted current responses. (b) Voltage responses.

Now consider driving a detailed parasitic network. The output currents from pre-characterization can be applied to the network at a given timestep. There will be a unique current that will elicit the same voltage on both a lumped capacitance and the network at the given timestep. This current is the chosen value for the given timestep, and this procedure is reapplied at every subsequent timestep. In other words, there is nothing more than application of $I_{out}(V_{out})(t)$.

CCS Timing delay calculation uses advanced interpolation technology to determine a current waveform when the input slew and/or output load values do not match those used during cell characterization. Additionally, interpolation is used for intermediate values of V_{dd} and temperature by using data from multiple libraries.

11.2.1.4 Characterization for CCS Timing

Characterizing cell timing for CCS Timing is very similar to characterization for NLDM: an input stimulus is chosen to produce a specific input slew time (S_{inp}); a load capacitance (C_{out}) is connected to the output pin; and a circuit simulation is run in the same way as NLDM. But instead of measuring voltage thresholds at the output pin, current is measured through the load capacitor and into the input pin. The current through C_{out} is used for the driver model, and the current into the input pin is used to determine the receiver model.

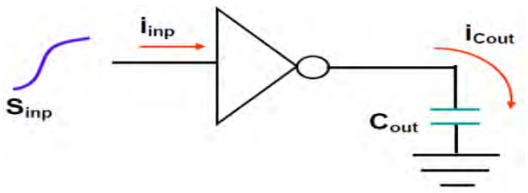


Figure 61: CCS Timing characterization measurements

These characterization experiments are repeated for a table of different S_{inp} and C_{out} combinations. The current through C_{out} is saved for every circuit simulation timestep and then reduced to a much smaller set of current and time (i, t) points. The points are chosen such that $V_{out}(t)$ can be accurately reproduced for every timestep during the transition. Figure 7 provides an example of the complete $i(t)$ waveform and a reduced set of points.

The current and voltage at the input pin are saved and then used to determine *C1* and *C2* values such that gate-level delay calculation can closely match times to the delay threshold and to the second slew threshold at the input pin. An additional piece of information, input reference time, is necessary in order to calculate cell delays. The reference time is the simulation time at which the waveform at the input pin crosses the rising or falling delay threshold (often this is 50% of V_{dd}).

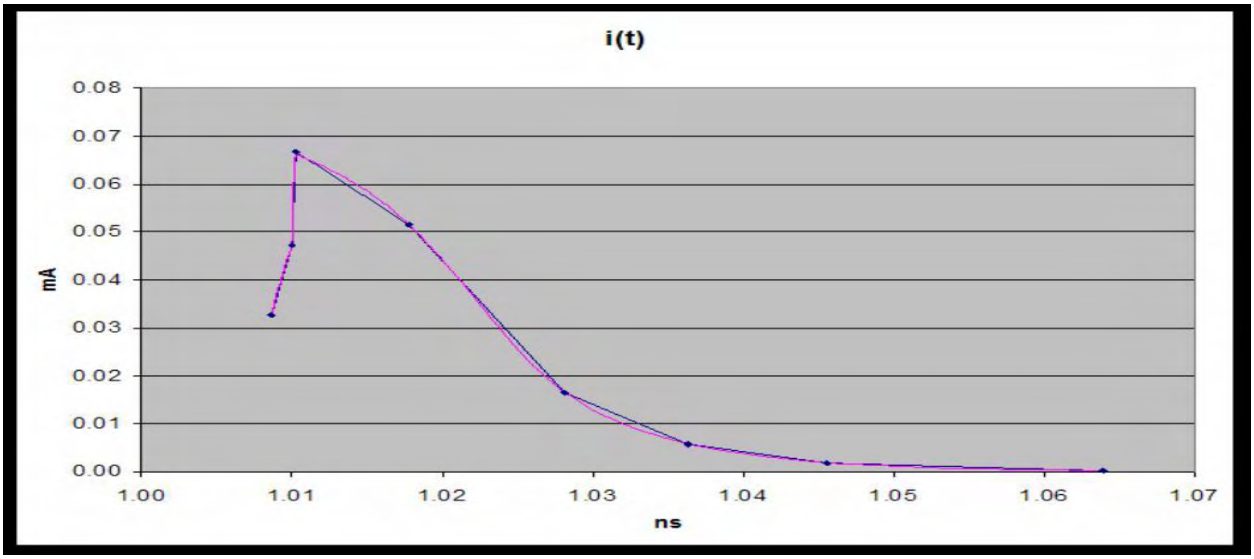


Figure 62: Current waveform from circuit simulation, and reduced current points

Representing CCS Timing Driver Information

In the Liberty syntax, using the CCS Timing model, you can represent nonlinear delay information at the pin level by specifying a lookup current table at the timing group level, dependent on input slew and output load [56].

CCS Timing Driver Lookup Table Model

You can represent CCS Timing driver models in your libraries by using lookup tables. To define your lookup tables, use the following groups and attributes:

• output_current_template group in the library group level

• output_current_rise and output_current_fall groups in the timing group level

Defining the output_current_template Group

Use this library-level group to create templates of common information that multiple current vectors can use. A table template specifies the CCS Timing driver model and the breakpoints for the axis. Specifying index_1, index_2, and index_3 values at the library level is optional.

11.2.1.4.1 Syntax

```
library(name_id) {
...
        output_current_template(template_nameid) {
                variable_1: input_net_transition ;
                variable_2: total_output_net_capacitance ;
                variable_3: time ;
                ...
        }
...}
```

11.2.1.4.2 Template Variables

The table template specifying CCS Timing driver models can have three variables (variable_1, variable_2, and variable_3). The valid values for variable_1 and variable_2 are input_net_transition and total_output_net_capacitance. The only valid value for variable_3 is time.

11.2.1.4.3 Example

```
library (new_lib) {
...
        output_current_template (CCT) {
                variable_1: input_net_transition;
                variable_2: total_output_net_capacitance;
                variable_3: time;
                ...
        }
        .. .
}
```

11.2.1.5 Defining the CCS Timing Output Currents

To specify the output current for the nonlinear table model, use the output_current_rise and output_current_fall groups within the timing group.

11.2.1.5.1 Syntax

```
timing() {
        output_current_rise () {
                vector (template_nameid){
                reference_time : float;
                index_1 (float);
                index_2 (float);
```

```
                index_3 ("float,..., float");
                values("float,..., float");
        }
}
```

11.2.1.6 Vector Group

Define the vector group in the output_current_rise or output_current_fall group. This group stores current information for a particular input slew and output load. There should be a minimum of four vector groups defined within a output_current_rise or output_current_fall group. In other words, current information should be specified for at least two different values of input slew and two different values of output load.

11.2.1.7 reference_time Simple Attribute

Define the reference_time simple attribute in the vector group. The reference_time attribute represents the time at which the input waveform crosses the rising or falling input delay threshold.

11.2.1.8 Template Variables

The table template specifying CCS Timing driver models can have three variables (variable_1, variable_2, and variable_3). The valid values for variable_1 and variable_2 are input_net_transition and total_output_net_capacitance. The only valid value for variable_3 is time. The index value for input_net_transition or total_output_net_capacitance is a single floating-point number. The index values for time are a list of floating-point numbers. The values attribute defines a list of floating point numbers that represent the current values of the driver model.

11.2.1.8.1 Example

```
library (nameid) {
...
        output_current_template (CCT) {
                variable_1: input_net_transition;
                variable_2: total_output_net_capacitance;
                variable_3: time;
        }
...
        timing() {
                output_current_rise() {
                        vector(CCT) {
                                reference_time : 0.05;
                                index_1(0.1);
                                index_2(2.1);
                                index_3("1.0, 1.5, 2.0, 2.5, 3.0");
                                values("1.1, 1.2, 1.5, 1.3, 0.5");
                        }
                        . ..
                }
                output_current_fall() {
                        vector(CCT) {
                                reference_time : 0.05;
                                index_1(0.1);
                                index_2(2.1);
```

```
                        index_3("1.0, 1.5, 2.0, 2.5, 3.0");
                        values("-1.1, -1.2, -1.5, -1.3, -0.5");
                    }
                    ...
                }
            }
}
```

The following rules apply for the CCS Timing driver model:

The variable lists of the output_current_template referred to in the vector group include only input_net_transition, total_output_net_capacitance, and time. Because the index_1, index_2, and index_3 variables are required in the lookup tables referring to output_current_template, in the output_current_template, these variables are optional. For half-unate sequential timing arcs, if timing_type is rising_edge or falling_edge and timing sense is positive_unate, only output_current_rise is required. If timing_type is rising_edge or falling_edge and timing sense is negative_unate, only output_current_fall is required. For all other timing arcs, output_current_rise and output_current_fall have to be defined in a pair. The output_current_rise and output_current_fall groups can coexist with other delay tables in the timing group. It is legal to define only output_current_rise or output_current_fall inside the timing group for half-unate arcs. If more than one output_current_rise or output_current_fall groups are defined in a timing group, only the last group defined counts. Inside each output_current_rise or output_current_fall group, the reference_time must be the same for each load of an input_net_transition value. In the vector group, indexes for input_net_transition and total_output_net_capacitance can define only one value each. That is, for every input_net_transition and total_output_net_capacitance pair, there must be a current time vector. The same value of load points should be specified for each slew.

11.2.2 Representing CCS Timing Receiver Information

CCS Timing receiver modeling must be used in conjunction with CCS Timing driver modeling. This model greatly improves the receiver model accuracy. In this approach, the capacitance is adjusted at the delay threshold. The capacitances used to model the receiver are dependent on input slew and output load.

11.2.2.1 CCS Timing Lookup Table Model

Library information for CCS Timing receiver modeling can be defined in the following ways:

• At the pin level inside the receiver_capacitance group by using the receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, and receiver_capacitance2_fall groups

• At the timing level by using the receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, and receiver_capacitance2_fall groups

Values for rise and fall can be defined at the pin level or at the timing level. The pin-level definition does not depend on output capacitance and is useful when there are no forward timing arcs.

11.2.2.2 Defining the Receiver Modeling Group at the Pin Level

You can define a receiver_capacitance group at the pin level. Use the receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise and receiver_capacitance2_fall groups.

11.2.2.3 Defining the lu_table_tamplate Group

Use this library-level group to create templates of common information that multiple lookup tables can use. A table template specifies the table parameters and the breakpoints for each axis. Assign each template a name so that lookup tables can refer to it.

11.2.2.4 lu_table_template Group

Define your lookup table templates in the library group.

11.2.2.4.1 Syntax

```
library(name_id) {
...
        lu_table_template(template_nameid) {
                variable_1: input_net_transition ;
                index_1 ("float,..., float");
                ...
        }
        ...
}
```

Template Variables

In the pin group, the table template specifying CCS Timing receiver models can have one variable (variable_1). The only valid value is input_net_transition. The index values in the index_1 attribute are a list of floating-point numbers. The values in the list must be in increasing order. There should be a minimum of two values specified for index_1 attribute. In other words, the receiver characteristics should be specified for at least two different values of input_net_transition.

11.2.2.4.2 Example

```
lu_table_template(LTT1) {
        variable_1: input_net_transition;
        index_1("0.1, 0.2, 0.3, 0.4");
}
```

11.2.2.5 Defining the receiver_capacitance Group

To specify the receiver capacitance for the nonlinear table model, use the receiver_capacitance group within the pin group.

Syntax for Pin Level

```
pin(name_id) {
        direction: input; /* or "inout" */
        receiver_capacitance() {
                receiver_capacitance1_rise(template_nameid) {
                        index_1("float,..., float"); /* optional */
                        values("float,..., float");
                }
                receiver_capacitance1_fall(template_nameid) {
                        index_1("float,..., float"); /* optional */
                        values("float,..., float");
                }
                receiver_capacitance2_rise(template_nameid) {
```

```
                index_1("float,..., float"); /* optional */
                values("float,..., float");
        }
        receiver_capacitance2_fall(template_nameid) {
                index_1("float,..., float"); /* optional */
                values("float,..., float");
        }
    }
}
```

Template Variables

In the pin group, the table template specifying receiver characteristics can have one variable (variable_1). The only valid value is input_net_transition. The index values in the index_1 attribute are a list of floating-point numbers. The values in the list must be in increasing order. There should be a minimum of two values specified for index_1 attribute. In other words, the capacitance of the receiver model should be specified for at least two different values of input_net_transition. The values attribute defines a list of floating-point numbers that represent the capacitance of the receiver model.

Example for Pin Level

```
pin(A) { /* pin-based receiver model*/
        direction : input;
        receiver_capacitance() {
                receiver_capacitance1_rise(LTT1) {
                        values("1.0, 4.1, 2.1, 3.0");
                }
                receiver_capacitance1_fall(LTT1) {
                        values("1.0, 3.2, 2.1, 4.0");
                }
                receiver_capacitance2_rise(LTT1) {
                        values("1.0, 4.1, 2.1, 3.0");
                }receiver_capacitance2_fall(LTT1) {
                        values("1.0, 3.2, 2.1, 4.0");
                }
        }
}
```

The following rules apply for the CCS Timing pin-level receiver model:

1. In the receiver_capacitance group, the lu_table_template referred to by receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, and receiver_capacitance2_fall contains only input_net_transition as its parameter.
2. If any of receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, or receiver_capacitance2_fall are defined in the receiver_capacitance group, then all of them have to be defined.
3. If more than one receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, or receiver_capacitance2_fall are defined in the receiver_capacitance group, only the last one counts.
4. The receiver_capacitance group applies only to input or inout pins.
5. Receiver model exclusive check: If input pin A defines the receiver_capacitance group, then no timing arc A->Y (attached under pin Y) can define receiver_capacitance1_rise, receiver_capacitance1_fall,

receiver_capacitance2_rise, and receiver_capacitance2_fall.

That is, either a pin-based or arc-based receiver_capacitance group can be specified, but not both.

11.2.2.6 Defining the Receiver Modeling Group at the Pin Level

At the timing level, you do not need to define the receiver_capacitance group. Define the receiver capacitance for the timing arcs by using only the receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, and receiver_capacitance2_fall groups.

Defining the lu_table_template Group

Use this library-level group to create templates of common information that multiple lookup tables can use. A table template specifies the table parameters and the breakpoints for each axis. Assign each template a name so that lookup tables can refer to it.

lu_table_template Group

11.2.2.6.1 Syntax

```
library(name_id) {
...
        lu_table_template(template_nameid) {
                variable_1: input_net_transition ;
                variable_2: total_output_net_capacitance ;
                index_1 ("float,..., float");
                index_2 ("float,..., float");
                ...
        }
...}
```

11.2.2.7 Template Variables

The table template specifying CCS Timing receiver models can have only two variables (variable_1 and variable_2). The parameters are the input transition time and the total output capacitance. The index values in the index_1 and index_2 attributes are a list of floating-point numbers. The values in the list must be in increasing order. There should be a minimum of two values specified for index_* attributes corresponding to input transition time. In other words, the capacitance of the receiver model should be specified for at least two different values of input transition time.

11.2.2.7.1 Example

```
lu_table_template(LTT2) {
        variable_1: input_net_transition;
        variable_2: total_output_net_capacitance ;
        index_1("0.1, 0.2");
        index_2("1.0, 2.0");
}
```

11.2.2.8 Defining the receiver_capacitance Groups

To specify the receiver capacitance for the nonlinear table model, use the receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise receiver_capacitance2_fall groups within the timing group.

11.2.2.8.1 Syntax for Timing Level

```
direction: output; /* or "inout" */
timing () {
        ...
        receiver_capacitance1_rise(template_nameid) {
                index_1("float,..., float"); /* optional */
                index_2("float,..., float"); /* optional */
                values("float,..., float");
        }
        receiver_capacitance1_fall(template_nameid) {
                index_1("float,..., float"); /* optional */
                index_2("float,..., float"); /* optional */
                values("float,..., float");
        }
        receiver_capacitance2_rise(template_nameid) {
                index_1("float,..., float"); /* optional */
                index_2("float,..., float"); /* optional */
                values("float,..., float");
        }
        receiver_capacitance2_fall(template_nameid) {
                index_1("float,..., float"); /* optional */
                index_2("float,..., float"); /* optional */
                values("float,..., float");
        }
…}
```

11.2.2.9 Template Variables

In the timing level, the table template specifying CCS Timing receiver models can have two variables (variable_1 and variable_2). The valid values for either variable are input_net_transition and total_output_net_capacitance. The index values in the index_1 and index_2 attributes are a list of floating-point numbers. The values in the list must be in increasing order. There should be a minimum of two values specified for index_* attributes corresponding to input_net_transition. In other words, the capacitance of the receiver model should be specified for at least two different values of input_net_transition. The values attribute defines a list of floating-point numbers that represent the capacitance for the receiver model.

11.2.2.9.1 Example at the timing level

```
timing() { /* timing arc-based receiver model*/
        ...
        related_pin : "B"
        receiver_capacitance1_rise(LTT2) {
                values("1.1, 4., 2.0, 3.2");
        }
        receiver_capacitance1_fall(LTT2) {
                values("1.0, 3.2, 4.0, 2.1");
        }
        receiver_capacitance2_rise(LTT2) {
                values("1.1, 4., 2.0, 3.2");
        }
```

```
        receiver_capacitance2_fall(LTT2) {
                values("1.0, 3.2, 4.0, 2.1");
        }
        . ..
}
```

The following rules apply for the CCS Timing arc-based receiver model:

1. In the timing group, the lu_table_template referred to by receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, and receiver_capacitance2_fall must contain input_net_transition and total_output_net_capacitance as its parameters.
2. If any of receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, or receiver_capacitance2_fall are defined in the timing group, then all of them have to be defined.
3. If more than one receiver_capacitance1_rise, receiver_capacitance1_fall, receiver_capacitance2_rise, or receiver_capacitance2_fall are defined in the timing group, only the last one counts.

11.2.3 CCS Timing Engine

The following algorithm describes the iterative method developed in order to calculate driver's arc delay and output slew of a cell which belongs to the stage (level) i, using receiver's c1/c2 capacitance values based on the CCS Timing model:

11.2.3.1 CCS Iterative Algorithm:

- Set c1/c2 receiver capacitances of driving pins of stage (i+1) to their NLDM values.
- Set c1/c2 receiver capacitances of driving pins of stage (i+2) to their NLDM values (fixed during iterations)
- do{

    1) Find the closer CCS output current waveforms (from Liberty file) for the current (input_transition, total_output_c1_load) & (input_transition, total_output_c2_load) breakpoints (combinations)

    2) Transform the closer CCS output current waveforms to voltage waveforms (in order to compute output slew and delay)

    3) Calculate the output voltage waveform for the current breakpoints using advanced interpolation techniques on the closer voltage waveforms

    4) Find the time points where output voltage crosses the low, high and delay thresholds (t_lower_threshold (30%), t_delay_threshold (50%) and t_upper_threshold (70%))

    5) Calculate the new c1/c2 values of (i+1) stage driving pins according to the receiver model of CCS Timing

}while (c1/c2 values not converged) // (|c1_new – c1_old| > eps || |c2_new – c2_old| > eps)

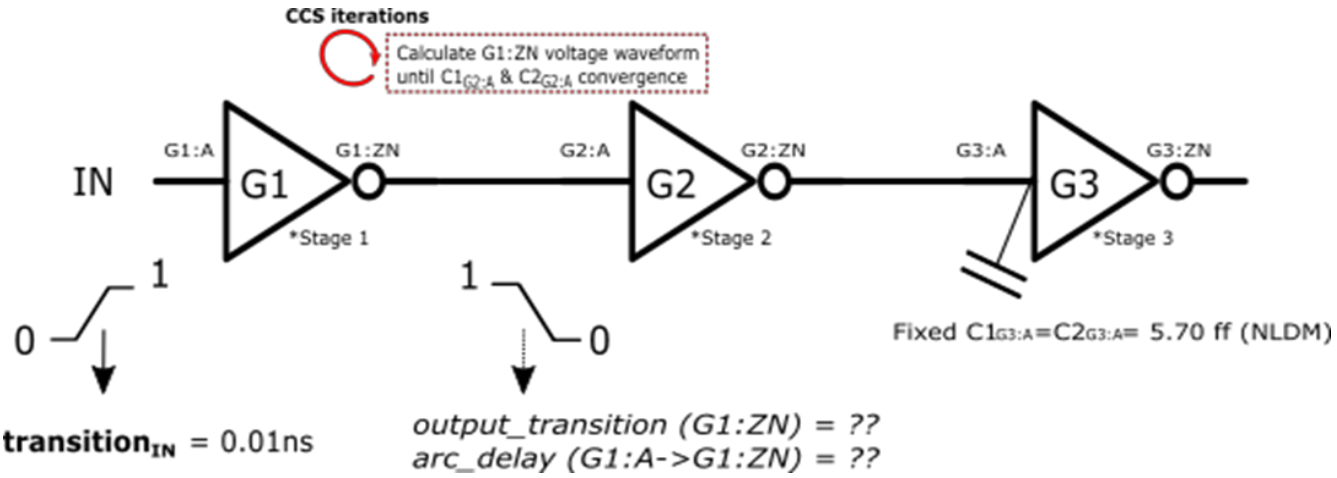
- Calculate the time point where the input voltage waveform crosses the delay threshold (t_reference) for the

current breakpoint

Output:
Arc_delay = t_delay_threshold - t_reference
Output_pin_slew = t_upper_threshold - t_lower_threshold

11.2.3.2 An example of the CCS Iterative Algorithm

Below is an example which demonstrates the calculation of a driver's CCS arc delay and output slew for an INV_X4 inverter of the Nangate-45nm library. Figure 63 shows the demonstration setup. Here, the goal is to calculate the CCS arc_delay ($_{G1:A(rise)}$ ⇨ $_{G1:ZN(fall)}$) delay and output_pin_slew ($_{G1:ZN(fall)}$) for the cell named G1 which belongs to stage 1. For simplicity we consider zero wire delay model and every wire capacitance is zero (C_{test} = C_{temp} = 0 ff). The absolute error ($|C_{new} - C_{old}|$) for the C1, C2 receiver capacitances in order to converge is set to 10^{-2}.



11.2.3.2.1 Calculation of CCS Timing Information for the G1 cell:

- **$C1_{G2:A}$** = **$C2_{G2:A}$** = $C_{G2:A(NLDM)}$ = 5.70 ff (they will change according to CCS model during the iterative algorithm)
- $C1_{G3:A}$ = $C2_{G3:A}$ = $C_{G3:A(NLDM)}$ = 5.70 ff (they will remain fixed)

⇨ **Iteration 1**

1. The closer CCS timing **output_current_fall** waveforms (4 vectors for C1 and the same 4 vectors for C2 capacitance) for input_transition/total_output_C1_capacitance & input_transition/total_output_C2_capacitance combinations ($transition_{IN}$ = 0.01ns & $C1_{G2:A}$ = $C2_{G2:A}$ = 5.70 ff) are the following:

Capacitance / Transition	0.365616	7.591250

0.0047239	vector(Output_current) {
	reference_time : 3.115910;
	index_1(0.00472397);
	index_2(0.365616);
	index_3("3.116950,3.117112,3.118412,3.119710,3.120956,
	3.121703,3.122460,3.123527,3.124121");
	values("-0.0287284,-0.0306436,-0.0565531,-0.0777294,-0.0931202,
	-0.0744040,-0.0402231,-0.0116323,-0.00577980"); }
		vector(Output_current) {
		reference_time : 3.115910;
		index_1(0.00472397);
		index_2(7.591250);
		index_3("3.117494,3.117731,3.119710,3.121794,3.122430,
		3.124519,3.126036,3.128000,3.131023");
		values("-0.369421,-0.386424,-0.689635,-0.981188,-1.091006,
		-0.953104,-0.631720,-0.263724,-0.0575905"); }

0.0171859	vector(Output_current) {
	reference_time : 3.131500;
	index_1(0.0171859);
	index_2(0.365616);
	index_3("3.126824,3.127834,3.130632,3.132990,3.135810,
	3.139067,3.140310,3.141717,3.143804");
	values("-0.00605896,-0.00646289,-0.0153340,-0.0247182,-0.0354299,
	-0.0448209,-0.0357565,-0.0119080,0.00172904"); }
		vector(Output_current) {
		reference_time : 3.131500;
		index_1(0.0171859);
		index_2(7.591250);
		index_3("3.128092,3.128994,3.132877,3.136426,3.140490,
		3.144513,3.146651,3.148907,3.152547");
		values("-0.0934096,-0.0996369,-0.236398,-0.376814,-0.518413,
		-0.608795,-0.464695,-0.198089,-0.0354414"); }

.
.
.

4. After transforming CCS current waveforms to voltage waveforms (step 2) and calculating the output voltage waveform for the current input_transition/total_output_C1_capacitance & input_transition/total_output_C2_capacitance combinations (step 3), we find the time points where output voltage crosses the low, high and delay thresholds:

t_lower_threshold = 3.127639 ns
t_delay_threshold = 3.129880 ns ⇨ **transition**$_{G1:ZN}$ = 0.004246 ns
t_upper_threshold = 3.131886 ns

5. In order to calculate the new values of $C1_{G2:A(CCS)}$ and $C2_{G2:A(CCS)}$ receiver capacitances, we use the **receiver_capacitance1_fall** for the (transition$_{G1:ZN}$ = 0. 004246 ns & $C1_{G3:A}$ = 5.70 ff) combination & the **receiver_capacitance2_fall** for the (transition$_{G1:ZN}$ = 0. 004246 ns & $C2_{G3:A}$ = 5.70 ff) combination.

receiver_capacitance1_fall

```
receiver_capacitance1_fall(Timing_7_7) {
        index_1 ("0.00117378,0.00472397,0.0171859,0.0409838,0.0780596,0.130081, 0.198535");
        index_2 ("0.365616,7.591250,15.182500,30.365000,60.730000,121.460000, 242.920000");
        values ("8.939820,9.151920,9.384120,9.564820,9.250800,9.293690,9.323600", \
                 "5.608010,5.569350,5.550280,5.545480,5.552470,5.565900, 5.573200", \
                 "5.902370,5.836540,5.800190,5.762830,5.727090,5.700690, 5.689880", \
                 "6.039080,5.960690,5.913490,5.860520,5.808000,5.764080, 5.733660", \
                 "6.125170,6.048300,5.998520,5.937900,5.875950,5.817680, 5.771450", \
                 "6.191670,6.116550,6.065870,6.003370,5.934630,5.867290, 5.809330", \
                "6.246570, 6.172400, 6.120480, 6.056710,5.986030,5.913500,5.846760");
}
```

receiver_capacitance2_fall

```
receiver_capacitance2_fall(Timing_7_7) {
        index_1("0.00117378,0.00472397,0.0171859,0.0409838,0.0780596,0.130081,0.198535");
        index_2 ("0.365616,7.591250,15.182500,30.365000,60.730000,121.460000,242.920000");
        values ("6.398590,6.482960,6.458690,6.601690,6.485170,6.538190,6.546860", \
```

```
                "6.063930,5.863270,5.795520,5.734700,5.686760,5.661930,5.653150", \
                "7.769190,6.541690,6.189300,5.928530,5.762540,5.665790,5.611580", \
                "10.431100,7.995940,7.140550,6.493220,6.074880,5.829390,5.692200", \
                "12.113900,9.801940,8.467410,7.325530,6.549140,6.087360,5.828190", \
                "12.372500,11.426700,9.951260,8.376920,7.179880,6.436350,6.015590", \
                "12.379900,12.203300,11.292200,9.539580,7.950230,6.876660,6.254470");
        }
```

Interpolating on those LUTs, we get the new C1,C2 receiver capacitances:

$C1_{G2:A}$ = 6.05 ff & **$C2_{G2:A}$** = 5.99 ff (NOT converged yet)

⇨ **Iteration 2**

1. The closer CCS timing current waveforms for the new ($transition_{IN}$, $C1_{G2:A}$) & ($transition_{IN}$, $C2_{G2:A}$) combinations remain the same.

   .
   .
   .

4. t_lower_threshold = 3.127740 ns
   t_delay_threshold = 3.130056 ns ⇨ **$transition_{G1:ZN}$** = 0.004289 ns
   t_upper_threshold = 3.132063 ns

5. **receiver_capacitance1_fall** for the ($transition_{G1:ZN}$ = 0.004289 ns & $C1_{G3:A}$ = 5.70 ff) combination & the **receiver_capacitance2_fall** for the ($transition_{G1:ZN}$ = 0. 004289 ns & $C2_{G3:A}$ = 5.70 ff) combination, remain the same.
   Interpolating on those LUTs, we get:

   $C1_{G2:A}$ = 6.01 ff & **$C2_{G2:A}$** = 5.98 ff (NOT converged yet)

⇨ **Iteration 3**

1. The closer CCS timing current waveforms for the new ($transition_{IN}$, $C1_{G2:A}$) & ($transition_{IN}$, $C2_{G2:A}$) combinations remain the same.

   .
   .
   .

4. t_lower_threshold = 3.127758 ns
   t_delay_threshold = 3.130035 ns ⇨ **$transition_{G1:ZN}$** = 0.004301 ns
   t_upper_threshold = 3.132059 ns

5. **receiver_capacitance1_fall** for the ($transition_{G1:ZN}$ = 0. 004301 ns & $C1_{G3:A}$ = 5.70 ff) combination & the **receiver_capacitance2_fall** for the ($transition_{G1:ZN}$ = 0. 004301 ns & $C2_{G3:A}$ = 5.70 ff) combination, remain the same.
   Interpolating on those LUTs, we get:

   $C1_{G2:A}$ = 5.99 ff & **$C2_{G2:A}$** = 5.98 ff (NOT converged yet)

⇨ **Iteration 4**

1. The closer CCS timing current waveforms for the new ($transition_{IN}$, $C1_{G2:A}$) & ($transition_{IN}$, $C2_{G2:A}$) combinations remain the same.

   .
   .
   .

4. t_lower_threshold = 3.127753 ns
   t_delay_threshold = 3.130028 ns ⇨ **$transition_{G1:ZN}$** = 0.004305 ns
   t_upper_threshold = 3.132058 ns

5. **receiver_capacitance1_fall** for the ($transition_{G1:ZN}$ = 0. 004305 ns & $C1_{G3:A}$ = 5.70 ff) combination & the **receiver_capacitance2_fall** for the ($transition_{G1:ZN}$ = 0. 004305 ns & $C2_{G3:A}$ = 5.70 ff) combination, remain the same.
   Interpolating on those LUTs, we get:
   $C1_{G2:A}$ = 5.99 ff & **$C2_{G2:A}$** = 5.98 ff (**converged**)

✓ **End of iterations**

- Using **$transition_{IN}$** to interpolate between the reference time values for the closer characterized transition times, we get:

Reference Time (ns)
3.115910
3.131500

⇨ t_reference = 3.122510 ns

Output:

arc_delay ($_{G1:A(rise)}$ ⇨ $_{G1:ZN(fall)}$) = 3.130028 - 3.122510 = **0.007518** ns
output_pin_slew ($_{G1:ZN(fall)}$) = t_upper_threshold - t_lower_threshold = 3.132058 - 3.127753 = **0.004305** ns

11.3 Appendix C

11.3.1 Even- and Odd-Mode Excitations

Symmetrical four-port networks are of particular interest since they allow for analysis in terms of even- and odd-mode excitation [43]. For simplicity, let's consider the symmetric four-port network of **Figure 61**, where all ports are terminated by the system impedance of Z_0. By using the S-parameter conventions, we can describe the network properties by using incident and reflected signals at all ports. Scattering parameters allow us to describe the network properties in terms of the incident and reflected signals as given by the following expression.

$$\begin{bmatrix} V_{1-} \\ V_{2-} \\ V_{3-} \\ V_{4-} \end{bmatrix} = \begin{bmatrix} S_{11}\, S_{12}\, S_{13}\, S_{14} \\ S_{21} S_{22} S_{23}\, S_{24} \\ S_{31} S_{32}\, S_{33}\, S_{34} \\ S_{41} S_{42}\, S_{43}\, S_{44} \end{bmatrix} \times \begin{bmatrix} V_{1+} \\ V_{2+} \\ V_{3+} \\ V_{4+} \end{bmatrix} \quad \text{(C.1)}$$

Due to symmetry and the reciprocal nature of the network we may state that $S_{ij} = S_{ji}$, where $i, j = 1 \ldots 4$ and $S11 = S33$, $S22 = S44$, $S34 = S12$, and $S23 = S14$, which results in a compact S-parameter matrix representation.

$$[S] = \begin{bmatrix} [S_A] & [S_B] \\ [S_B] & [S_A] \end{bmatrix} \quad \text{(C.2)}$$

$$[S_A] = \begin{bmatrix} S_{11} & S_{21} \\ S_{21} & S_{22} \end{bmatrix} \quad \text{(C.3)}$$

$$[S_B] = \begin{bmatrix} S_{31} & S_{41} \\ S_{41} & S_{42} \end{bmatrix} \quad \text{(C.4)}$$

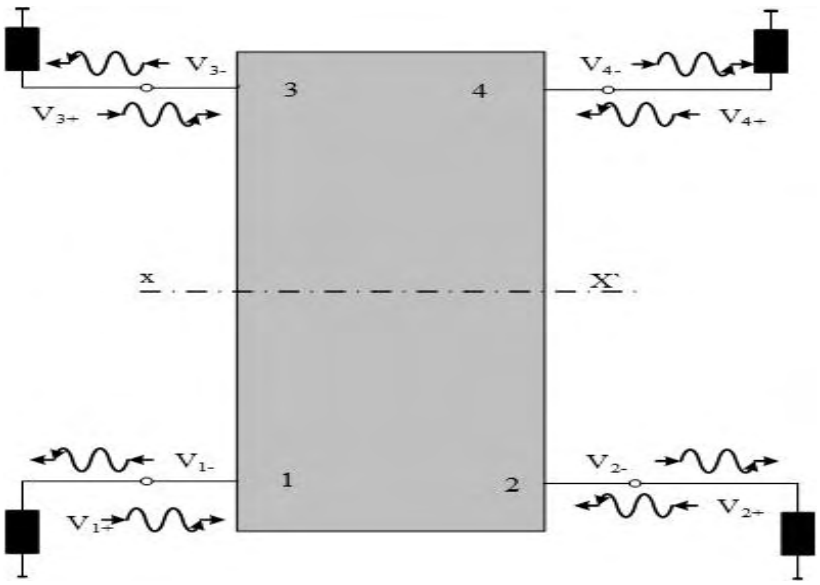


Figure 64 Symmetric four-port network.

Substituting the compact matrix formulation back to (C.1) we obtain a matrix representation.

$$\begin{bmatrix} V_{1-} \\ V_{2-} \\ V_{3-} \\ V_{4-} \end{bmatrix} = \begin{bmatrix} [S_A] & [S_B] \\ [S_B] & [S_A] \end{bmatrix} \times \begin{bmatrix} V_{1+} \\ V_{2+} \\ V_{3+} \\ V_{4+} \end{bmatrix} \quad \text{(C.5)}$$

It is interesting to note what conclusions can be drawn from the symmetry of the network. Applying in-phase signals of equal magnitude at ports 1 and 3 results in corresponding in-phase voltages of equal magnitude at those ports. Applying the same excitation scheme for ports 2 and 4 will cause the same result for the port voltages. This scheme is known as even-mode excitation. However, applying signals of equal magnitude that are out of phase will result in what is called odd-mode excitation.

11.3.2 Even-Mode Analysis

By applying the even-mode excitation to our four-port network we obtain the circuit representation of **Figure 62**. The symmetry plane $x - x\_$ corresponds now to an open-circuit (OC). Let $V1\pm = V3\pm = V1e\pm$ and $V2\pm = V4\pm = V2e\pm$be the even-mode signals to be considered in the analysis. Using this naming convention and (C.5) allows us to formulate the even-mode matrices [43].

$$\begin{bmatrix} V_{1e-} \\ V_{2e-} \\ V_{1e-} \\ V_{2e-} \end{bmatrix} = \begin{bmatrix} [S_A] & [S_B] \\ [S_B] & [S_A] \end{bmatrix} \times \begin{bmatrix} V_{1e+} \\ V_{2e+} \\ V_{1e+} \\ V_{2e+} \end{bmatrix} \quad \text{(C.6)}$$

$$\begin{bmatrix} V_{1e-} \\ V_{2e-} \end{bmatrix} = ([S_A] + [S_B]) \times \begin{bmatrix} V_{1e+} \\ V_{2e+} \end{bmatrix} \quad \text{(C.7)}$$

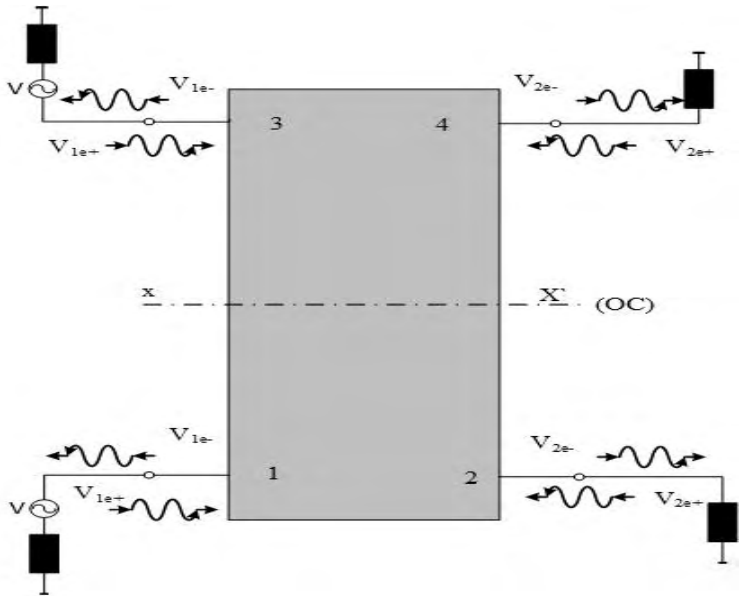


Figure 65 Symmetric four-port network.

11.3.3 Odd-Mode Analysis

The corresponding odd-mode excitation scheme with $V1\pm = -V3\pm = V1o\pm$ and $V2\pm = -V4\pm = V2o\pm$ and its short-circuit (SC) symmetry plane $x - x\_$ can be seen in **Figure 63**. Following a similar chain of calculations, as performed for the even-mode analysis, we may rewrite the network matrix equations and obtain the odd-mode matrices [43].

$$\begin{bmatrix} V_{1o-} \\ V_{2o-} \\ -V_{1o-} \\ -V_{2o-} \end{bmatrix} = \begin{bmatrix} [S_A] & [S_B] \\ [S_B] & [S_A] \end{bmatrix} \times \begin{bmatrix} V_{1o+} \\ V_{2o+} \\ -V_{1o+} \\ -V_{2o+} \end{bmatrix} \quad \text{(C.8)}$$

$$\begin{bmatrix} V_{1o-} \\ V_{2o-} \end{bmatrix} = ([S_A] - [S_B]) \times \begin{bmatrix} V_{1o+} \\ V_{2o+} \end{bmatrix} \quad \text{(C.9)}$$

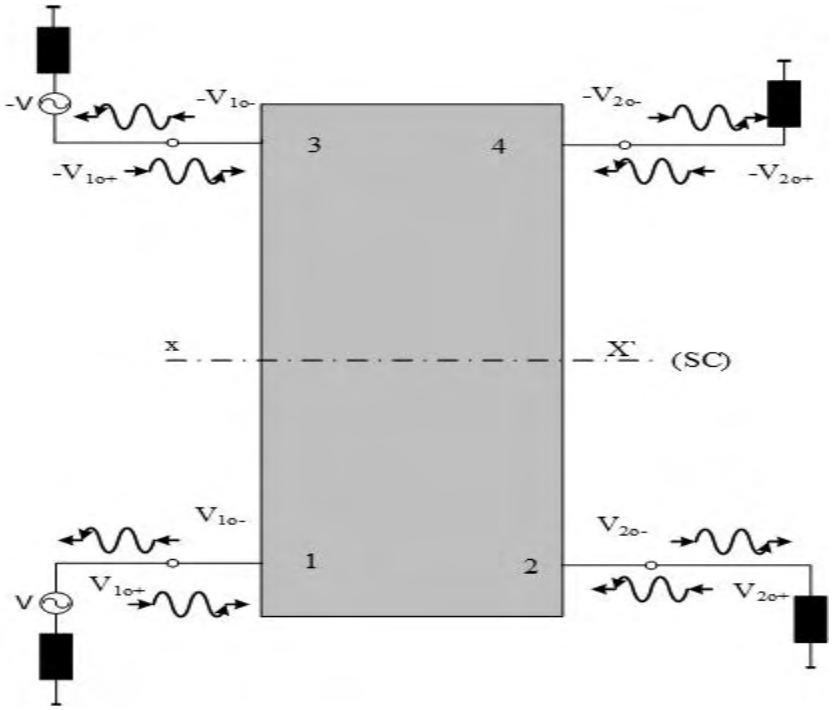


Figure 66 Symmetric four-port network.

11.4 Appendix D

STA (PrimeTime®) - Benchmark: C1908					
STA (1.1V)		**STA (0.9V)**		**IR-drop annotated**	
N4 (in)	0.00r	N43 (in)	0.00r	N43 (in)	0.00f
NOT1_2	0.00r	NOT1_15	0.00r	NOT1_15	0.01r
BUFF1_82	0.00f	BUFF1_86	0.01f	BUFF1_86	0.02r
BUFF1_220	0.02f	BUFF1_224	0.02f	BUFF1_223	0.04r
NOT1_272	0.03f	NOT1_280	0.04f	NOT1_278	0.04f
NAND2_314	0.04r	NAND2_318	0.05r	NAND2_317	0.05r
NAND2_336	0.05f	NAND2_340	0.06f	NAND2_339	0.06f
NOT1_367	0.06r	NOT1_375	0.07r	NOT1_377	0.07r
NAND2_396	0.06f	NAND2_402	0.07f	NAND2_403	0.08f
NAND2_412	0.07r	NAND2_416	0.08r	NAND2_417	0.09r
NOT1_428	0.08f	NOT1_429	0.09f	BUFF1_437	0.11r
BUFF1_457	0.09r	BUFF1_450	0.10r	BUFF1_475	0.12r
NOT1_479	0.10r	BUFF1_493	0.12r	NOT1_504	0.13f
NAND2_497	0.11f	NOT1_515	0.13r	NAND2_519	0.14r
NAND2_510	0.11r	NAND2_528	0.14f	NAND2_530	0.15f
NOT1_535	0.13f	NAND2_542	0.15r	NOT1_548	0.16r
NAND2_544	0.13r	NOT1_560	0.16f	NAND2_562	0.17f
NAND2_558	0.14f	NAND2_570	0.17r	NAND2_572	0.18r
BUFF1_588	0.15r	NAND2_583	0.18f	NOT1_589	0.19f
NOT1_604	0.17r	NOT1_603	0.19r	NOT1_606	0.19r
NAND2_621	0.17f	NOT1_622	0.20f	NAND2_624	0.20f
NAND2_633	0.18r	NAND2_634	0.21r	NAND2_635	0.21r

NOT1_641	0.19f	NAND2_642	0.22f	NAND2_643	0.22f
NAND2_649	0.19r	NAND2_650	0.22r	BUFF1_657	0.24f
BUFF1_659	0.20f	BUFF1_661	0.24f	NOT1_666	0.25r
NOT1_670	0.22f	NOT1_674	0.25f	NAND2_676	0.26f
NAND2_678	0.22r	NAND2_680	0.26r	NAND2_682	0.27r
NAND2_684	0.23f	NAND2_686	0.27f	BUFF1_691	0.28r
AND2_693	0.24r	AND2_695	0.28r	AND2_703	0.32r
BUFF1_698	0.26r	BUFF1_700	0.31r	NAND5_730_A	0.35f
NAND5_714_A	0.29r	NAND5_730	0.34r	NAND5_730	0.36r
NAND5_714	0.32f	NAND4_899	0.37f	AND8_793_A	0.41r
AND8_793_A	0.33r	AND8_793_A	0.38r	AND8_793_B	0.44r
AND8_793_B	0.36r	AND8_793_B	0.43r	AND8_793	0.46r
AND8_793	0.40r	AND8_793	0.46r	NAND2_822	0.48f
NAND2_822	0.42r	NAND2_822	0.49r	AND3_255	0.51f
AND3_836	0.44f	AND3_836	0.51f	NAND2_850	0.52r
NAND2_850	0.46f	NAND2_850	0.53f	NAND2_868	0.53f
NAND2_868	0.46r	NAND2_868	0.54r	NOT1_844	0.54r
NOT1_877	0.48f	NOT1_877	0.56f	NAND4_56	0.55f
NAND2_878	0.48r	NAND2_878	0.56r	OR2_879	0.56r
NAND2_879	0.49f	NAND2_879	0.58f	AND3_56	0.57r
AND2_880	0.50r	AND2_880	0.58r	**N3442 (out)**	**0.57r**
N2899 (out)	**0.52r**	**N2899 (out)**	**0.60r**		

12 References

[1] J. C. Butcher, Numerical Methods for Ordinary Differential Equations. Wiley, 2008.

[2] Y. Saad, Iterative Methods for Sparse Linear Systems. SIAM, 2003.

[3] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Up-date/Downdate," ACM Trans. Math. Softw., vol. 35, no. 3, pp. 22:1–22:14, 2008.

[4] https://www.sciencedirect.com/topics/engineering/electromigration

[5] https://overclocking.guide/the-risks-of-overclocking/

[6] https://www.cambridge.org/core/journals/journal-of-materials-research/article/thermomigration-of-cusn-and-nisn-intermetallic-compounds-during-electromigration-in-pbfree-snag-solder-joints/311B719F682B3679B52F9FC8213B84E1

[7] http://www.vlsi-expert.com

[8] Texas Instruments Inc., "Timing closure for system on a chip using voltage drop based standard delay formats", US Patents 7324914, Jan 29, 2008.

[9] Sebastian Dietel, Sebastian Hoppner,Holger Eisenreich, Georg Ellguth, Stefan Hanzsche,Stephan Henker and Ren´e Schuffny and Tim Brauninger, *"A Compact on-Chip IR-Drop Measurement System in 28 nm CMOS Technology"*, in Circuits and Systems (ISCAS), June 2014.

[10]Apple Inc., "*IR (voltage) drop analysis in integrated circuit timing"*, US Patent 8712752 B2, Apr 29, 2014.

[11] Y.-M. Jiang and K.-T. Cheng, "*Analysis of Performance Impact Caused by Power Supply Noise in Deep Submicron Devices*", in Proceedings of DAC, pages 760-765, June 1999.

[12] J. Galambos, "*The Asymptotic Theory of Extreme Order Statistics*", *2nd ed.*, Krieger, 1987.

[13] Bao Liu and Lu Wang, *"Dynamic Statistical-Timing-Analysis-Based VLSI Path Delay Test Pattern Generation"*, in IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2014.

[14] Angela Krstic, Yi-Min Jiang and Kwang-Ting (Tim) Cheng, "*Pattern Generation for Delay Testing and Dynamic Timing Analysis Considering Power-Supply Noise Effects*", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2001.

[15] Jing-Jia Liou, Angela Krstic, Yi-Min Jiang and Kwang-Ting Cheng, "*Path Selection and Pattern Generation for Dynamic Timing Analysis Considering Power Supply Noise Effects*", in IEEE ICCAD 2000.

[16] Hari Cherupalli and John Sartori, "*Graph-based Dynamic Analysis: Efficient Characterization of Dynamic Timing and Activity Distributions*", in IEEE ICCAD 2015.

[17] Vishweshwara R, Udayakumar H, Venkatraman R, Arvind N V, *"An Approach to Measure the Performance Impact of Dynamic Voltage Fluctuations Using Static Timing Analysis"*, in International Conference on VLSI Design 2009.

[18] B. Lasbouygues, R. Wilson, N. Azemard, P. Maurine, *"Temperature and Voltage Aware Timing Analysis: Application to Voltage Drops"*, in Design, Automation & Test in Europe Conference 2007.

[19] Yi-Min Jiang, Angela Krstic, Kwang-Ting (Tim) Cheng, "*Dynamic Timing Analysis Considering Power Noise Effects*", in ISQED 2000.

[20] M.K. Tsiampas, D. Bountas, P. Merakos, N.E. Evmorfopoulos, S. Bantas and G.I. Stamoulis, "*A Power Grid Analysis and Verification Tool Based on a Statistical Prediction Engine*", in IEEE ICECS, 2010.

[21] Dimitris Bountas, George Stamoulis, Anand Raman, "*A High-Capacity Power Integrity Flow Supporting Inductive Rail Effects with Transistor-Level Accuracy.*", Nasdaq –GlobeNewswire, March 2010.

[22] N. Evmorfopoulos, G. Stamoulis, and J. Avaritsiotis, "*A Monte Carlo approach for maximum power estimation based on extreme value theory*", *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 415-432, 2002.

[23] S. Resnick, "*Extreme Values, Regular Variation and Point Processes*", Springer, 1987.

[24] E. Castillo, "*Extreme Value Theory in Engineering*", Academic Press, 1988.

[25] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*, Dover, 1964.

[26] D. Luenberger, Linear and Nonlinear Programming, 2nd ed., Addison-Wesley, 1984.

[27] R. Fletcher, Practical Methods of Optimization, 2nd ed., Wiley, 1987.

[28] NanGate Free PDK45 Open Cell Library http://www.nangate.com/?page_id=2325

[29] Synopsys, PrimeTime User Guide, 2014.

[30] Microsoft Research, Z3 theorem prover http://rise4fun.com/z3/tutorial

[31] Beirlant J., J.Teugels, Yu.Goegebeur, J.Segers "Statistics of Extremes".

[32] L. de Haan and A. Ferreira, "*Extreme Value Theory: An Introduction"*, Springer, 2006.

[33] M. Falk, J, Husler, and R. Reiss, "*Laws of Small Numbers: Extremes and Rare Events"*, 3rd ed., Birkhauser, 2011.

[34] Kotz, S. and Nadarajah, S. (2000) "Extreme Value Distributions: Theory and Applications." Imperial College Press, London.

[35] Coles (2001), "An Introduction to Statistical Modeling of Extreme Values".

[36] Drees (2001), "Minimax Risk Bounds in Extreme Value Theory".

[37] Reiss and Thomas (2001), "Statistical Analysis of Extreme Values".

[38] Fougères (2004), “Extreme Values in Finance, Telecommunications, and the Environment”.

[39] E. Gumbel, “Bivariate exponential distributions”, J. American Statistical Association, vol. 55, pp. 698-707, 1960.

[40] J. Tawn, Modelling multivariate extreme value distributions, Biometrika, vol. 77, pp. 245-253, 1990.

[41] A. Ledford and J. Tawn, “Statistics for near independence in multivariate extreme values”, *Biometrika*, vol. 83, pp. 169-187, 1996.

[42] Errikos Lourandakis , Konstantinos Nikellis, Michail Tsiampas, Shinji Yamaura, *Member, IEEE,* and Yuu Watanabe, “Parametric Analysis and Design Guidelines for mm-Wave Transmission Lines in nm CMOS”, IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES 2018.

[43] Errikos Lourandakis, “*On-Wafer_Microwave_Measurements_and_De-embedding*”, 2016.

[44] Pozar, D. M., *Microwave Engineering*, New York: JohnWiley & Sons, 2009.

[45] Hirota, T., Minakawa, A., and Muraguchi, M., “Reduced-Size Branch-Line and Rat-Race Hybrids for Uniplanar MMIC’s,” *IEEE Transactions on Microwave Theory and Techniques*, Vol. 38, No. 3, 1990, pp. 270–275.

[46] Vecchi, F., et al., “Design of Low-Loss Transmission Lines in Scaled CMOS by Accurate Electromagnetic Simulations,” *IEEE Journal of Solid-State Circuits*, Vol. 44, No. 9, 2009, pp. 2605–2615.

[47] Zhang,W., et al., “Equivalent CircuitModeling of SlowWave Coplanar Strips forMillimeter Wave Applications,” *Proc. Int. Wireless Symposium*, Xian, China, March 24–26, 2014, pp. 1–4.

[48] Yang, B., E. Skafidas, and R. J. Evans, “A Novel Slow-Wave Structure for Millimeter-Wave Filter Application on Bulk CMOS,” *Proc. Radio and Wireless Symposium*, Phoenix, AZ, January 16–19, 2011, pp. 138–141.

[49] Lugo-Alvarez, J., et al., “High-Directivity Compact Slow-Wave Coplanar Waveguide Couplers for Millimeter-Wave Applications,” *Proc. 44th European Microwave Conference*, Rome, Italy, October 6–9, 2014, pp. 1072–1075.

[50] Verona, B., et al., “Slow-Wave Distributed MEMS Phase Shifter in CMOS for Millimeter-Wave Applications,” *Proc. 44th European Microwave Conference*, Rome, Italy, October 6–9, 2014, pp. 211–214.

[51] Ponchak, G. E., J. Papapolymerou, and M. M. Tentzeris, “Excitation of Coupled Slotline Mode in Finite-Ground CPW with Unequal Ground-Plane Widths,” *IEEE Transactions on Microwave Theory and Techniques*, Vol. 53, No. 2, 2005, pp. 713–717.

[52] Thukydides Xanthopoulos, Clocking in modern VLSI Systems.

[53] Victor H. Cordero and Sunil P Khatri “Clock Distribution Scheme using Coplanar Transmission Lines”.

[54] Sajjad Moazeni, “High-Frequency Clock Distribution Methods in Digital Integrated Circuits”.

[55] Rafael Escovar and Robert Suaya, “Transmission line design of Clock Trees”.

[56] www.eecs.berkeley.edu

[57] studylib.net

[58] Synopsys Inc., CCS Timing Technical White Paper version 2.0, 2006.

[59] Synopsys Inc., CCS Power Technical White Paper version 3.0, 2006.

[60] Nanotropic S.A. “System and method for determining simulated response extrema for integrated circuit power supply Network”, US Patent 8516423, August 20, 2013.

[61] http://www.vlsiencyclopedia.com/2011/06/dynamic-timing-analysis.html